Security of the Blockchain against Long Delay Attack

Puwen Wei[1(✉)], Quan Yuan[1(✉)], and Yuliang Zheng[2]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
pwei@sdu.edu.cn, yuanquan_sdu@mail.sdu.edu.cn
[2] University of Alabama at Birmingham, Birmingham, USA
yzheng@uab.edu

Abstract. The consensus protocol underlying Bitcoin (the blockchain) works remarkably well in practice. However proving its security in a formal setting has been an elusive goal. A recent analytical result by Pass, Seeman and shelat indicates that an idealized blockchain is indeed secure against attacks in an asynchronous network where messages are maliciously delayed by at most $\Delta \ll 1/np$, with n being the number of miners and p the mining hardness. This paper improves upon the result by showing that if appropriate inconsistency tolerance is allowed the blockchain can withstand even more powerful external attacks in the honest miner setting. Specifically we prove that the blockchain is secure against long delay attacks with $\Delta \geq 1/np$ in an asynchronous network.

Keywords: Bitcoin · Blockchain · Delay · Random oracle

1 Introduction

Bitcoin introduced by Nakamoto [19] is the first cryptocurrency that allows a ledger to be maintained by the public in a decentralized manner. It has a number of attractive properties including decentralization and pseudonymity. At the core of Bitcoin is a consensus protocol, called the blockchain. The blockchain is a chain-structured ledger maintained by all the participants (or miners), where records (or blocks) can only be added by the miners to the end of the chain.

A key idea of Nakamoto's blockchain protocol to achieve consensus among distributed miners is the use of proof of work (POW), which requires the miners to solve a "cryptographic puzzle". Advantages of POW are two folds. First, the "cryptographic puzzle" makes it more difficult for an adversary to modify the block. Second, POW helps distributed miners to synchronize in a permissionless setting. While having low efficiency and high power consumption, the blockchain protocol based on POW is still the most successful one that gains peoples acceptance wildly in practice. The main concern over the blockchain protocol based on

POW is security, which has not been proven formally until Garay, Kiayias, and Leonardas [10] provide a rigorous analysis of the blockchain protocol. They model the execution of the blockchain protocol by allowing the adversary to control a concrete percentage of computing power and also to interfere with communication among miners, whereby proving that two basic properties, which are common prefix and chain quality, hold for a blockchain built on POW. Considering the effect of delay, Pass, Seeman and shelat [22] prove the security of the blockchain protocol in an asynchronous network with a-priori bounded delay Δ, where the adversary can delay any message with at most Δ rounds. The security analysis in [22] holds for a relatively small delay only. Specifically the delay Δ should be significantly smaller than $1/np$, that is $\Delta \ll 1/np$, where n and p denote the number of miners and the mining hardness, respectively.

Networks delay is considered to be one of the most important threats to the security of a blockchain. As shown in [6], long delays lead to increased probabilities for forking, which may break the common prefix property. Pass, Seeman and shelat demonstrate a simple attack in a fully asynchronous setting where the adversary is allowed to schedule message delivery with a long delay relative to the mining hardness. What is worse, such attacks could be deployed even when all miners are honest, which means that the adversary does not need any hashing power [10].

In the real world, however, long delays, say $\Delta \geq 1/np$, could be caused not only by message propagation over a "bad" asynchronous network but also by malicious attacks. Instead of attempting to corrupt a sizable fraction of miners, it would be much easier for the adversary to disrupt communications among miners. Furthermore, it is also unpractical to require all the miners' chains to be consistent with the "main chain" due to the long delay.

In practice the adversary cannot delay messages successfully all the time. Consider the eclipse attack [14] that allow an adversary to control 32 IP addresses to monopolize all connections to and from a target bitcoin node with 85% probability. If the attack fails or the adversary loses the ability to intercept messages, blocks will be diffused to other miners at an exponentially fast rate. This naturally brings up a interesting question, that is

Is the blockchain protocol based on POW still secure in a real world asynchronous network, where long delay relative to the mining hardness, say $\Delta \geq 1/np$, is allowed?

Our contribution. In this paper, we focus on the effect of long delay, especially $\Delta \geq 1/np$, and give results that support a positive answer to the above question. Specifically, we propose a simplified model for the blockchain protocol based on POW, which captures an adversary's ability to deliver messages maliciously in the real world. We extend the definitions of chain growth and common prefix [10][22][11] to allow fractions of miners' chains to be inconsistent with the main chain. By analyzing the evolution of the main chain in a more subtle way, we prove that the common prefix property and the chain growth property still hold in our model. In addition, to illustrate the threat of long delay attack in

our model, we present a concrete attack in which an adversary without any hash power may threaten the common prefix property of a blockchain protocol with certain parameters.

There are a number of subtle differences between our model and previous research in [10][22][11]. A detailed discussion follows.

- Long delay attack: In our model, the upper bound of delay can be large, say $\Delta \geq 1/np$, and the adversary can delay a message with probability $\alpha \in (0, 1]$, meaning that the adversary may not always disrupt communications successfully in practice. Previous works consider $\Delta \ll 1/np$ or $\Delta = 1$ and the adversary can always delay any message. Hence, our model is more general in capturing the adversary's ability to deliver messages maliciously.
- Common prefix for majority: We relax the requirements of common prefix and chain growth so that certain fractions of miners' chains are allowed to be inconsistent with the common prefix of the main chain. Previous definitions of common prefix require all the miners' chains be consistent with the common prefix of the main chain, which is a special case of our definition. We emphasize that such inconsistency tolerance is not only crucial to our proof but also necessary for the blockchain protocol to work in practice.
- Honest miners: Since we only focus on the effect of delay, we assume all the miners are honest. That is, all the miners follow the protocol honestly and the adversary neither corrupts any miners nor possesses any hash power. Hence, we only need to consider the common prefix property and the chain growth property. Previous works consider adversaries which can collect a fraction of the total hash power by means of corrupting miners and thus analyze chain quality. Additionally we impose restrictions on the miners' behavior: two consecutive blocks cannot be mined by the same miner. This restriction is reasonable in our honest miner setting, as in practice is unlikely that two consecutive blocks are mined by the same miner[3], especially when n is large whereas p is small.

In a large-scale blockchain protocol, it is hard for the adversary to collect enough computational power to mount an effective attack, where at least 1/3 computational power of all miners is usually required. Therefore, we ignore the influence of the hash power of an adversary and instead focus on attacks by disrupting communications.

Main techniques. Informally, the common prefix property states that, in addition to the last T blocks, all the miners' chains should have the same prefix. In order to prove the common prefix property, [22] shows that there are enough "convergence opportunities" for the miners to synchronize the same chain, where the "convergence opportunities" depend on consecutive Δ rounds of "silence". Here, Δ rounds of "silence" means no honest miners mines a block during these Δ rounds. If $\Delta \ll 1/np$, it is likely that no block is mined during Δ rounds. However, the challenge is that, if $\Delta \geq 1/np$, at least one block is expected to

[3] not the same mining pool

be mined during those rounds, which will ruin the "convergence opportunities". So previous proof techniques cannot be applied when $\Delta \geq 1/np$. To solve this problem, we introduce an inconsistency tolerance parameter λ, which is inspired by the fact that the common prefix property in the real world holds only for the majority miners. Therefore, we redefine the properties of chain growth rate and common prefix using $\lambda \in (\frac{1}{2}, 1]$, which captures to what extent the common prefix property holds. Our definitions are more general and allow us to exclude the "bad" miners during Δ rounds of silence. Furthermore, we introduce a powerful tool called $\mathsf{Tree}_{\mathsf{MC}}$ to record the state of the main chains. Unlike the $\mathcal{F}_{tree}$ oracle in [22] which stores all the chains during the execution of the blockchain protocol, our $\mathsf{Tree}_{\mathsf{MC}}$ only records the state of the main chain at the current round, which can capture the evolution of main chains in a subtle manner. Then, we show the relation between $\mathsf{Tree}_{\mathsf{MC}}$ and the view of the real execution of blockchain protocol. Due to the good properties of $\mathsf{Tree}_{\mathsf{MC}}$, we only need to focus on the analysis of $\mathsf{Tree}_{\mathsf{MC}}$ instead of the original block chain protocol, which greatly simplified the analysis and security proof.

Related Work. Since the introduction of Bitcoin, a number of cryptocurrency, e.g., Litecoin, Zerocash [2] and Ethereum, have appeared, most of which are based on the idea of Bitcoin. Meanwhile, a series of works [26][6][8][31][29][9][3][28][27][30][21][12][15][23] analyze the security of Bitcoin under different attack scenarios and investigate the conditions under which Bitcoin achieves a game-theoretic equilibrium. Eyal and Sirer [8] propose an attack strategy called "selfish mining", where the adversary only requires about 1/3 of the total mining power. Miller and LaViola [18] show the connection between bitcoin and probabilistic Byzantine agreement protocols. Heilman et al. [14] present eclipse attacks which allow an adversary controlling a sufficient number of IP addresses to "eclipse" a bitcoin node. As mentioned in [14], the attacker can eclipse a fraction of miners and launch N-confirmation double spending attacks without any mining power. In fact, such attacks can be extended to attacks on common prefix. For instance, the attacker can eclipse a fraction of miners in advance and launch the long delay attacks described in section 7. Notice that the target block which the attacker intends to delay may be not mined by the eclipsed miners. In other words, a block can be delayed with some probability, which is the scenario captured by our model. Sompolinsky and Zohar [29] show that the bitcoin protocol with high throughput is more susceptible to double-spend attacks. In order to solve the above problem, [29] presents an algorithm called GHOST, which chooses the main chain by the heaviest subtree instead of the longest branch. Then, Natoli and Gramoli [20] propose the balance attack against POW blockchain systems, where the common prefix property can be broken by disrupting communications between subgroups of similar mining power.

Rigorous cryptographic analysis on blockchain protocol are initiated by Garay, Kiayias and Leonardas [10] and Pass, Seeman and shelat [22]. [10] abstracts the backbone protocol of Bitcoin and proves its security under the proposed model. Furthermore, [22] extends the model to an asynchronous network and shows the

security of blockchain protocol with a bounded delay $\Delta \ll 1/np$. Kiayias and Panagiotakos [16] investigate the tradeoff between provable security and transaction processing speed. Then, Garay, Kiayias and Leonardas [11] analyze the security of blockchain protocol with variable difficulty. Pass and Shi [24] consider the sleepy model, where players can be either online or offline. Notice that it is difficult for the adversary to control large fractions of the total mining power in practice, and no such attacks has been observed to date. Hence, Badertscher et al. [1] investigates the reason why we can assume the majority of the mining power is honest or why the miners need to follow the protocol honestly. In order to overcome the problems induced by POW, such as large energy demands, another line of research focuses on the blockchain protocol based on proof of stake (POS), where the miner to issue the next block is decided by randomly selecting one of the miners proportionally to their stakes. For instance, Algorand [13], Snow White [4], Ouroboros/Ouroboros Praos [17][5], and Thunderella [25].

2 Preliminaries

In this section, we recall the blockchain protocol, following the notations of [10][22].

2.1 Notation

Let B denote a block. A blockchain $C = \overrightarrow{B}$ consists of a sequence of ordered Bs and the length $|C|$ means the number of blocks in C. Let m denote the message contained in B. $\overrightarrow{m}$ denotes the messages in $\overrightarrow{B}$ correspondingly. We denote by C_i^r the chain of miner i at round r. $C^{\lfloor k}$ denotes the chain C that removes the last k blocks, where k is a nonnegative integer. If $k \geq |C|$, $C^{\lfloor k} = \varepsilon$. Let $C_1 \preceq C_2$ denotes that C_1 is a prefix of C_2. $\mathsf{B}(n, k)$ denotes the binomial distribution with n trials and success probability k. $H : \{0,1\}^* \to \{0,1\}^\kappa$ is a cryptographic hash function.

2.2 Blockchain protocol

A blockchain protocol consists of two algorithms, which are Π^V and $\mathcal{C}$. Π^V is a stateful algorithm, receiving security parameter κ and maintaining a blockchain C. C is a sequence of block B, where $B = (h_{-1}, m, r, h)$. h_{-1} is a pointer to the previous block. m is the message from the environment. r is a nonce. h is the pointer to the current block such that $h = H(h_{-1}, m, r)$. The cryptographic hash function $H(\cdot)$ is modeled by a random oracle $\mathsf{H}(\cdot)$, which on inputs x outputs $H(x)$. Let $\mathsf{H.ver}(\cdot, \cdot)$ be an oracle which takes (x, y) as inputs and outputs 1 if $H(x) = y$ and 0 otherwise. The first block of a chain is called the genesis block $B_0 = (0, \perp, 0, H(0||\perp||0))$. The algorithm $\mathcal{C}$ takes C as input and outputs the corresponding sequence of messages $\overrightarrow{m}$ of C. That is, $\mathcal{C}(C) = \overrightarrow{m}$. V is an algorithm which checks the validity of $\overrightarrow{m}$. If $\overrightarrow{m}$ is valid, $V(\mathcal{C}(C))$ outputs 1. In

the bitcoin protocol, m contains the transaction information and V is used to check the validity of transactions.

A block $B = (h_{-1}, m, r, h)$ is valid with respect to a predecessor block $B_{-1} = (h'_{-1}, m', r', h')$ only if following conditions hold:

- $h_{-1} = h'$,
- $h = H(h_{-1}, m, r) < D_p$, where D_p is the difficulty parameter.

If all blocks in C are valid and $V(\mathcal{C}(C)) = 1$, we say C is valid, where the corresponding validity check algorithm is called "chain-check" algorithm.

Suppose there are n miners, where $n = n(\kappa)$ is a polynomial function with κ. At each round, a miner receives a message m from the environment Z and runs Π^V to maintain a chain C as follows:

- If $V(\mathcal{C}(C)||m) \neq 1$, proceed to the next step. Otherwise, pick $r \leftarrow \{0,1\}^\kappa$ randomly and compute h by querying H with (h_{-1}, m, r), where h_{-1} is the pointer of the last block of C. If $h < D_p$, set $C = CB$, where $B = (h_{-1}, m, r, h)$, and we say the miner succeeds in mining a new block B. The miner can query H at most q times before he succeeds. Then, broadcast the new chain C. In order to capture the attack that the adversary can disturb the communication among miners, C is considered as being delivered by the adversary.
- On receiving the chains delivered by the adversary, choose the longest and valid one, say C', where the validity of blocks is checked by querying $\mathsf{H.ver}$. If $|C'| > |C|$, replace C by C'. Otherwise, go to the next round.

Note that under the random oracle model $H(\cdot)$ is modeled by a random oracle $\mathsf{H}(\cdot)$ and a miner is allowed to query H for at most q times at each round, but can query $\mathsf{H.ver}$ for arbitrary times. p denotes the probability that a miner succeeds in mining a block at a round, where $p = 1 - (1 - \frac{D_p}{2^\kappa})^q \approx \frac{qD_p}{2^\kappa}$. We use p to describe the difficulty of mining in the following parts.

2.3 $\mathcal{F}_{tree}$ model

In this section we recall the simplified blockchain protocol with access to $\mathcal{F}_{tree}$ oracle introduced by [22]. The $\mathcal{F}_{tree}$ oracle maintains a tree which contains messages of all valid chains and can answer two kinds of queries, $\mathsf{Tree.extend}$ and $\mathsf{Tree.ver}$. When receiving query $\mathsf{Tree.extend}((B_0, ..., B_{l-1}), B)$, it checks whether $(B_0, ..., B_{l-1})$ is a path of the tree, where the root of the tree is the genesis block B_0. If so, with probability p it extends this path with B and returns 1; Otherwise, return 0. When receiving $\mathsf{Tree.ver}(B_0, ..., B_l)$, it returns 1 if $(B_0, ..., B_l)$ is a path of the tree; Otherwise, return 0. Here, a block B only contains message m, i.e., $B = (m)$. Then the random oracle in blockchain protocol is replaced with $\mathcal{F}_{tree}$ and the resulting protocol is called $(\Pi_{tree}, \mathcal{C}_{tree})$. The main differences between $(\Pi_{tree}, \mathcal{C}_{tree})$ and $(\Pi, \mathcal{C})$ are described as follows.

The protocol $(\Pi_{tree}, \mathcal{C}_{tree})$ is also directed by an environment $Z(1^\kappa)$. The environment activates n miners and sends each miner a message at each round. A miner receives a message m from the environment Z and runs Π_{tree} below:

- If $V_{tree}(\mathcal{C}_{tree}(C)||m) \neq 1$, proceed to the next step. Otherwise, query $\mathcal{F}_{tree}$ with $\mathsf{Tree.extend}(C, m)$. If the oracle answers 1, a new block $B = (m)$ is mined. Set $C = CB$ and broadcast C.
- When receiving the chains delivered by the adversary, choose the longest and valid one, say C', where the validity of C' can be checked by querying $\mathsf{Tree.ver}(C')$. If the oracle $\mathsf{Tree.ver}(C')$ returns 1, we say the chain is valid. If $|C'| > |C|$, set $C = C'$. Otherwise, go to the next round.

Under the $\mathcal{F}_{tree}$ model, a miner is allowed to query $\mathsf{Tree.extend}$ only once at each round, but can query $\mathsf{Tree.ver}$ for arbitrary times. Note that the miners described in section 2.3 can query H at most q times at a round and the probability of successful mining at a round is p. Therefore those queries to H at a round are considered as one query to $\mathsf{Tree.extend}$.

[22] shows that the security properties in $(\Pi_{tree}, \mathcal{C}_{tree})$ still hold in original protocol, while the analysis is much simpler in the $\mathcal{F}_{tree}$ model. For simplicity, we misuse $(\Pi, \mathcal{C})$ to denote the basic blockchain protocol in the $\mathcal{F}_{tree}$ model. Besides, the algorithm V_{tree} or V depends on the functionality of the concrete protocol. To simplify the description, we consider V which outputs 1 for all inputs. Hence, V is omitted in following parts.

3 Blockchain Model with Long Delays

Nakamoto's blockchain protocol is proved to be secure [22], where chains broadcasted by miners may suffer at most Δ-bounded delays such that $\Delta \ll 1/np$. As discussed in Section 1, on one hand, it is much easier for the adversary to disturb the communications rather than collect large computational power. On the other hand, if the adversary fails to delay the target chain, the chain will be diffused to other miners immediately. To capture such scenario, our modifications for the behavior of the adversary are as below:

Execution of adversary at round r:

- **Recieving**. On receiving the chains from miners, the adversary chooses which valid chains he wants to delay. But only with probability α the chosen one can be delayed. Those delayed chains are marked as "delayable" . The other undelayed chains are marked as "undelayable". Then all the chains the adversary received together with their marks and the round r are saved in a list $\mathcal{T}$.
- **Distribution**. The adversary chooses which chains in $\mathcal{T}$ to be distributed and these chains will be received by all the miners at the next round. But the following two kinds of chains have to be distributed at the current round.
  - The chains marked as undelayable;
  - The chains having been marked as delayable for Δ rounds.

Note that if the adversary distributes more than one chains at a round, the adversary can adjust the order of these chains. For instance, the adversary can

broadcast chains C_1 and C_2 in a way that (C_1, C_2) to miner i but (C_2, C_1) to miner j, where $|C_1| = |C_2|$. If C_1 and C_2 are longer than i and j's chains, then i accepts C_1 as his main chain while j accepts C_2. We emphasize that our model is in honest miner setting where the adversary does not corrupt any miners.

Remark. In practice, it is possible that some miners receive a block earlier than others due to the propagation delay in the bitcoin network. As shown in [DW13], the broadcast of a block follows an exponential behavior. Hence, once a block has been broadcasted to its neighbors, most miners will receive the block immediately and it is difficult for the adversary to delay anymore. It takes about 10 seconds for a broadcasted block to be known by almost all the miners [DW13][PSS17]. In our model, if the adversary broadcasts a block, all the miners will receive it in the next round, where the time span of a round can be 10 seconds. Such time span is enough for the adversary to influence the miners' behavior. To capture the possible attacks, e.g., attacks on miner i and j described above, we allow the adversary can adjust the order of these chains, which is equivalent to the case that miner i received C_1 only.

Modification to blockchain protocol. We make additional restrictions on the miners' behavior in the blockchain protocol. That is, the miner cannot mine in a chain, the last block of which was mined by himself. In other words, the miner who has already mined a block will not execute the mining step of Π until he receives a new chain mined by other miners. The reason why we prevent consecutive blocks mined by the same miner is that such consecutive blocks may cause possible forks even in the honest miner setting. In addition, it is not likely that a miner (not the mining pool) can mine two consecutive blocks in practice due to the large number of miners n and the small difficulty parameter p. Hence, such a restriction is reasonable in our honest miner setting.

We emphasize that our restriction only applies to a single miner which is an independent communication node of the network and has a unit computational power. Hence, such a modification would lead to a slightly decline of the total mining power and we ignore such a mild change in the following proof for simplicity.

In our protocol, we say a miner is "being delayed" if his chain is being delayed by the adversary. Obviously, a miner being delayed will not mine a block untill he accepts a new chain mined by others.

4 Properties of Our Blockchain Model

In this section we redefine the chain growth property and the common prefix property in our blockchain model.

4.1 Chain growth

Previous definition of chain growth [22] considers the minimum increase of the length of all miners' chains during T rounds. In our model, we consider the

length increase of the majority of miners' chains instead. Informally speaking, if the majority of chains, say, with fraction $\lambda > \frac{1}{2}$, grows by t blocks during consecutive rounds, we say the blockchain view grows by t blocks during these rounds with majority λ. In fact, the definition of chain growth in [22] is a special case of ours when $\lambda = 1$. It is, however, difficult to have $\lambda = 1$ in practice. Hence, our definition is more flexible in capturing the real scenario.

Let $\texttt{view}(\Pi, \mathcal{C}, A, Z, \kappa)$ and $|\texttt{view}(\Pi, \mathcal{C}, A, Z, \kappa)|$ denote the joint view of all miners and the number of rounds during the execution of $(\Pi, \mathcal{C})$, respectively.

Definition 1. *Given* $\texttt{view}(\Pi, \mathcal{C}, A, Z, \kappa)$*, we say the blockchain grows by at least* t *blocks with majority* $\lambda \in (\frac{1}{2}, 1]$ *from round* r_1 *to* r_2*, if*

$$\Pr_{i,j}[|C_j^{r_2}| - |C_i^{r_1}| \geq t] \geq \lambda, \tag{1}$$

where the probability is taken over all the choice of $i, j \in [n]$*.*

Define $\texttt{chain-increase}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r_1, r_2, \lambda)$ *as the maximum value of* t *satisfying (1). That is,*

$$\texttt{chain-increase}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r_1, r_2, \lambda) = \mathit{max}\{t | \Pr_{i,j}[|C_j^{r_2}| - |C_i^{r_1}| \geq t] \geq \lambda\}.$$

Definition 2. *The blockchain protocol* $(\Pi, \mathcal{C})$ *has the chain growth rate* $g \in \mathbb{R}$ *with majority* $\lambda \in (\frac{1}{2}, 1]$*, if there exists some constant* c *and negligible functions* ϵ_1, ϵ_2 *such that for every* $\kappa \in \mathbb{N}, T \geq c\log(\kappa)$ *and every* $r \leq |\texttt{view}(\Pi, C, A, Z, \kappa)| - T$ *, the following holds:*

$$\Pr[\texttt{chain-increase}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r, r+T, \lambda) \geq gT] \geq 1 - \epsilon_1(\kappa) - \epsilon_2(T), \tag{2}$$

where the probability is taken over the randomness of the protocol.

4.2 Common prefix

Similarly, we can define common prefix as follows.

Definition 3. $\texttt{common-prefix}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r, k, \lambda) = 1$ *with majority* $\lambda \in (\frac{1}{2}, 1]$ *if the following holds:*

$$\Pr_{i,j}[(C_i^{r\lfloor k} \preceq C_j^r) \wedge (C_j^{r\lfloor k} \preceq C_i^r)] \geq \lambda, \tag{3}$$

where the probability is taken over all the choice of $i, j \in [n]$*.*

Definition 4. *A blockchain protocol* $(\Pi, \mathcal{C})$ *satisfies the common prefix property with parameter* $\lambda \in (\frac{1}{2}, 1]$*, if there exists some constant* c *and negligible function* ϵ_1 *and* ϵ_2 *such that for every* $\kappa \in \mathbb{N}, T \geq c\log(\kappa)$ *and every* $r \leq |\texttt{view}(\Pi, C, A, Z, \kappa)|$ *, the following holds:*

$$\Pr[\texttt{common-prefix}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r, T, \lambda) = 1] \geq 1 - \epsilon_1(\kappa) - \epsilon_2(T), \tag{4}$$

where the probability is taken over the randomness of the protocol.

5 State of the Main Chain

In this section, we introduce a special tree to capture the evolution of the main chains.

5.1 Record the state of the main chain

During the execution of Π, miners will "reach agreement" on some chains at each round and those chains are called the main chains. Although the main chains may not be unique at each round, only one of those chains will be the prefix of the main chain after enough rounds. Since the evolution of the main chains is closely related to chain growth and common prefix, we introduce a special tree, denoted by $\mathsf{Tree}_{\mathsf{MC}}$, to record the state of the main chains, where a node of the tree is a block of a chain. $\mathsf{Tree}_{\mathsf{MC}}$ is initialized to the root B_0. Next, we show how to add and delete blocks at a round in $\mathsf{Tree}_{\mathsf{MC}}$.

- **AddBlock**: When the adversary broadcasts a chain $C = (B_0, B_1, ..., B_l)$, and there exist a branch (or paths from root to leaves) C' in $\mathsf{Tree}_{\mathsf{MC}}$ such that $C' = C^{\lfloor k}$ with the smallest k, extend C' with the last k ordered blocks of C. Note that the adversary is allowed to send more than one chain at a round. That means the same leaf node of $\mathsf{Tree}_{\mathsf{MC}}$ may be extended with different branches simultaneously.
- **DeleteBlock**: At the end of a round (after the adversary finishes **Distribution**), suppose $\mathsf{Tree}_{\mathsf{MC}}$ has the depth, say d. Delete "useless" blocks or forks so that only the branches Cs satisfying the following conditions remain.
  - $|C| = d$,
  - For any C' with depth d, the last block of C was added to $\mathsf{Tree}_{\mathsf{MC}}$ no later than the last block of C'.

Once the adversary broadcast the chains, each miner will update his chain with the longer one, and no one will withhold the shorter chains or attempt to extend them. Hence, $\mathsf{Tree}_{\mathsf{MC}}$ only records all the main chains of the undelayed miners at current round. But if a miner has a chain longer than the main chain but is delayed by the adversary, this delayed chain is not recorded in $\mathsf{Tree}_{\mathsf{MC}}$.

5.2 Properties of $\mathsf{Tree}_{\mathsf{MC}}$

Obviously, all of the branches on $\mathsf{Tree}_{\mathsf{MC}}$ at the end of a round are of equal depth and the depth of $\mathsf{Tree}_{\mathsf{MC}}$ never decreases. Other interesting properties of $\mathsf{Tree}_{\mathsf{MC}}$ are shown below.

Lemma 1. *Properties of $\mathsf{Tree}_{\mathsf{MC}}$.*

1. *If new blocks are successfully added to $\mathsf{Tree}_{\mathsf{MC}}$ at the end of a round, then the depth of $\mathsf{Tree}_{\mathsf{MC}}$ increases.*
2. *The depth of $\mathsf{Tree}_{\mathsf{MC}}$ increases by at most 1 at each round.*

3. *If only one block is added to* $\mathsf{Tree}_{\mathsf{MC}}$ *at the end of a round, then* $\mathsf{Tree}_{\mathsf{MC}}$ *has only one branch and the depth increases by 1.*

Proof. 1. Suppose there are new blocks added to the $\mathsf{Tree}_{\mathsf{MC}}$ while the depth remains unchanged. So those added blocks are useless and will be deleted at once due to **DeleteBlock**.
2. Without loss of generality, suppose the depth of $\mathsf{Tree}_{\mathsf{MC}}$ at round $r-1$ and r are d and $d+2$, respectively. If the $(d+2)$th block is mined by miner i, then $(d+1)$th block must be mined by a different miner, say miner j, due to the restriction that the same miner cannot mine two consecutive blocks. Hence, miner i received miner j's chain of length $d+1$ from the adversary. That means there exists a round r' such that $r' < r$ and the depth of $\mathsf{Tree}_{\mathsf{MC}}$ is $d+1$ at round r', which contradicts the fact that the depth of $\mathsf{Tree}_{\mathsf{MC}}$ is d at round $r-1$.
3. Suppose the depth of $\mathsf{Tree}_{\mathsf{MC}}$ is d at round r and only one block, say B, is successfully added at round $r+1$. Due to the first property, the depth increases to $d+1$. And the length of branches without B is still d. After **DeleteBlock**, the useless blocks of these branches will be deleted from the tree and only the branch with depth $d+1$ will remain.

5.3 Relation with the view of $(\Pi, \mathcal{C})$

$\mathsf{Tree}_{\mathsf{MC}}$ records the main chains known by all the miners at current round. But there are some chains at current round which are not recorded in $\mathsf{Tree}_{\mathsf{MC}}$ due to the adversarial delay. Hence, the actual view of the main chains of $(\Pi, \mathcal{C})$ may be different from $\mathsf{Tree}_{\mathsf{MC}}$. Fortunately, such difference in terms of chain growth and common prefix is negligible. Therefore, we can prove these properties of $(\Pi, \mathcal{C})$ by analyzing $\mathsf{Tree}_{\mathsf{MC}}$. The relations between $\mathsf{Tree}_{\mathsf{MC}}$ and the view of $(\Pi, \mathcal{C})$ are proven by the following lemmas. (Note that the following lemmas are all discussed after $\mathsf{Tree}_{\mathsf{MC}}$ finishes the step of **DeleteBlock**.)

Lemma 2. *Assume* $1/2 < \lambda \leq 1 - 8\alpha p\Delta$. *Let* m^r_{delay} *be the number of being delayed miners at round* r. *There exists a polynomial function poly such that*

$$\Pr[m^r_{delay} > \frac{(1-\lambda)n}{4}] < e^{-poly(\kappa)}. \tag{5}$$

Proof. Consider the case that $r \geq \Delta$. If a miner i is being delayed at round r, that means i succeeded in mining a delayable block from round $r-\Delta+1$ to round r. During these Δ rounds, there are $n\Delta$ independent events of mining, each of which is delayable with probability αp. So $m^r_{delay} \sim \mathsf{B}(n\Delta, \alpha p)$. According to the Chernoff bound, for any $\epsilon \geq 1$, we have

$$\Pr[m^r_{delay} > (1+\epsilon)\alpha np\Delta] < e^{\frac{-\epsilon\alpha np\Delta}{3}}. \tag{6}$$

Let $(1+\epsilon)\alpha np\Delta = \frac{(1-\lambda)n}{4}$ and $1/2 < \lambda \leq 1-8\alpha p\Delta$. We have $\epsilon = \frac{1-\lambda}{4\alpha p\Delta} - 1 \geq 1$. Therefore,

$$\Pr[m^r_{delay} > \frac{(1-\lambda)n}{4}] < e^{\frac{-\epsilon(1-\lambda)n}{12(1+\epsilon)}} \leq e^{\frac{-(1-\lambda)n}{24}}, \tag{7}$$

where the last inequality follows from $\frac{\epsilon}{1+\epsilon} \geq \frac{1}{2}$. Since $n = n(\kappa)$ is a polynomial function with κ, let $poly(\kappa) = \frac{(1-\lambda)n(\kappa)}{24}$. That completes the proof of Lemma 2.

We denote the event that $m^r_{delay} > \frac{(1-\lambda)n}{4}$ as **Over-delay** in the following parts.

Lemma 3. *Assume $1/2 < \lambda \leq 1 - 8\alpha p\Delta$. Let d^r_{tree} be the depth of $\mathsf{Tree}_{\mathsf{MC}}$ at round r. We have*

$$\Pr[\texttt{chain-increase}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r_1, r_2, \lambda) \geq d^{r_2}_{tree} - d^{r_1}_{tree}] \geq 1 - 2e^{-poly(\kappa)}. \tag{8}$$

Proof. If $|C^r_i| < d^r_{tree}$ which means there exists at least one chain of length d^r_{tree} distributed by the adversary and known to all the miners, miner i at the end of round r should have updated his state with the chain of length d^r_{tree}. That is, $|C^r_i| = d^r_{tree}$. So the event that $|C^r_i| < d^r_{tree}$ cannot happen.

If $|C^r_i| > d^r_{tree}$, which means C^r_i is being delayed by the adversary. Assuming that **Over-delay** doesn't happen at round r_1 and r_2 (with probability at least $1 - 2e^{-poly(\kappa)}$ due to Lemma 2), we have

$$\Pr_i[|C^r_i| \neq d^r_{tree}] = \frac{m^r_{delay}}{n} \leq \frac{1-\lambda}{4}. \tag{9}$$

Therefore,

$$\begin{aligned}
&\Pr_{i,j}[|C^{r_2}_j| - |C^{r_1}_i| \geq d^{r_2}_{tree} - d^{r_1}_{tree}] \\
\geq\ &\Pr_{i,j}[|C^{r_2}_j| - |C^{r_1}_i| = d^{r_2}_{tree} - d^{r_1}_{tree}] \\
\geq\ &1 - \Pr_i[|C^{r_1}_i| \neq d^{r_1}_{tree}] - \Pr_j[|C^{r_2}_j| \neq d^{r_2}_{tree}] \\
\geq\ &1 - \frac{1-\lambda}{4} - \frac{1-\lambda}{4} > \lambda.
\end{aligned}$$

That means $\texttt{chain-increase}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r_1, r_2, \lambda) \geq d^{r_2}_{tree} - d^{r_1}_{tree}$, which completes the proof of the Lemma 3.

Lemma 4. *Assume $1/2 < \lambda \leq 1 - 8\alpha p\Delta$. Let d be the depth of $\mathsf{Tree}_{\mathsf{MC}}$. If all the branches of $\mathsf{Tree}_{\mathsf{MC}}$ at round r have a common prefix with length $d - T$, we have*

$$\Pr[\texttt{common-prefix}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r, T, \lambda) = 1] \geq 1 - 2e^{-poly(\kappa)}. \tag{10}$$

Proof. Suppose all the branches of $\mathsf{Tree}_{\mathsf{MC}}$ at round r have a common prefix with length $d - T$. For any two branches of $\mathsf{Tree}_{\mathsf{MC}}$ at round r, say $C^r_{tree.1}$ and $C^r_{tree.2}$, we have $(C^r_{tree.1}{}^{\lfloor T} \preceq C^r_{tree.2}) \wedge (C^r_{tree.2}{}^{\lfloor T} \preceq C^r_{tree.1})$. However, not every miner's view match with $\mathsf{Tree}_{\mathsf{MC}}$. Suppose C^r_i is not a branch of $\mathsf{Tree}_{\mathsf{MC}}$ at round r, which is denoted by $C^r_i \notin \mathsf{Tree}^r_{\mathsf{MC}}$. Consider the following two cases:

- Case 1: C_r^i is being delayed by the adversary at round r. Assume that **Over-delay** doesn't happen at round r. As is discussed in the proof of lemma 3, the probability of this case is at most $\frac{1-\lambda}{4}$.
- Case 2: C_i^r is not being delayed by adversary at round r. Then $|C_i^r| = d_{tree}^r$. Suppose $C_i^r \notin \mathsf{Tree}_{\mathsf{MC}}^r$. That is, C_i^r has been distributed by the adversary, which means the last block of C_i^r was added to $\mathsf{Tree}_{\mathsf{MC}}$ due to **AddBlock** but then deleted due to **DeleteBlock** at round $r' \leq r$. Since $d_{tree}^{r'} \geq |C_i^r|$ and $d_{tree}^{r'} \leq d_{tree}^r$ due to Lemma 1, we have $d_{tree}^{r'} = |C_i^r| = d_{tree}^r$. Hence, there exists another branch C^* such that $|C_{tree}^*| = d_{tree}^r$ and C_{tree}^* is added to $\mathsf{Tree}_{\mathsf{MC}}$ earlier than C_i^r. Let r^* denote the round at which C_{tree}^* is added. Since C_{tree}^* is distributed by the adversary at round r^* but the miner i didn't update his state with C_{tree}^*, $C_i^{r^*}$ must be no shorter than C_{tree}^*. Therefore, $|C_i^{r^*}| = |C_i^r| = d_{tree}^r$ and $C_i^{r^*} = C_i^r$. We thus conclude that C_i^r was created no later than r^* but was distributed at round $r' > r^*$. That means, miner i was being delayed at round r^*.
  Assuming that **Over-delay** doesn't happen at round r^*, the probability that miner i was being delayed at round r^* is at most $\frac{1-\lambda}{4}$ due to the proof of Lemma 3. So the probability of $C_i^r \notin \mathsf{Tree}_{\mathsf{MC}}^r$ in this case is at most $\frac{1-\lambda}{4}$.

To sum up, on condition that **Over-delay** doesn't happen at round r^* and r (with probability at least $1 - 2e^{poly(\kappa)}$), the probability that C_i^r is not a branch of $\mathsf{Tree}_{\mathsf{MC}}$ at round r is

$$\begin{aligned}
&\Pr_i[C_i^r \notin \mathsf{Tree}_{\mathsf{MC}}^r] \\
=\ &\Pr_i[C_i^r \notin \mathsf{Tree}_{\mathsf{MC}}^r \wedge \text{Case 1}] + \Pr_i[C_i^r \notin \mathsf{Tree}_{\mathsf{MC}}^r \wedge \text{Case 2}] \\
\leq\ &\frac{m_{delay}^r}{n} + \frac{m_{delay}^{r^*}}{n} \\
\leq\ &\frac{1-\lambda}{4} + \frac{1-\lambda}{4} = \frac{1-\lambda}{2}
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\Pr_{i,j}[(C_i^r{\lfloor}^T \preceq C_j^r) \wedge (C_j^r{\lfloor}^T \preceq C_i^r)] \\
\geq\ &\Pr_{i,j}[C_i^r \subset \mathsf{Tree}_{\mathsf{MC}}^r \wedge C_j^r \subset \mathsf{Tree}_{\mathsf{MC}}^r] \\
\geq\ &1 - \Pr_i[C_i^r \notin \mathsf{Tree}_{\mathsf{MC}}^r] - \Pr_j[C_j^r \notin \mathsf{Tree}_{\mathsf{MC}}^r] \\
\geq\ &1 - \frac{1-\lambda}{2} - \frac{1-\lambda}{2} = \lambda,
\end{aligned}$$

which completes the proof of Lemma 4.

6 Proofs of Security

In this section we analyze the chain growth property and the common prefix property of $(\Pi, \mathcal{C})$ using $\mathsf{Tree}_{\mathsf{MC}}$.

6.1 Chain Growth

Theorem 1. *(Chain growth). Assume* $1/2 < \lambda \leq 1 - 8\alpha p\Delta$. *The blockchain protocol* $(\Pi, \mathcal{C})$ *has the chain growth rate* $g = \frac{(1-\delta)f}{1+fE[R^i_{delay}]}$ *with majority* λ, *where* $f = 1-(1-p)^n$, $E[R^i_{delay}] = \frac{\alpha - \alpha\omega^{\Delta-1}[\omega+\Delta(1-\omega^2)]}{1-\omega}$ *and* $\omega = 1-(1-\alpha)f$.

Proof. The aim of the adversary is to decrease the chain growth rate by delaying or scheduling the chain delivery. Due to Lemma 3, which shows the relation between the chain growth of (Π, C) and that of $\mathsf{Tree}_{\mathsf{MC}}$, we only need to focus on the chain growth of $\mathsf{Tree}_{\mathsf{MC}}$.

It seems that the adversary can use forks to distract the hashing power of miners in order to slow the chain growth rate. However, the forks does not help in breaking the chain growth property of $\mathsf{Tree}_{\mathsf{MC}}$. More precisely, consider the rounds at which two consecutive blocks are added to $\mathsf{Tree}_{\mathsf{MC}}$. Once a miner successfully mined a block B_1, which corresponds to chain C_1, the adversary can delay it with probability α for at most Δ rounds, and waits for the next block B'_1. If B_1 is delayable and the next block B'_1 corresponding to C'_1 is mined within Δ rounds, the adversary can generate a fork by broadcasting both chain C_1 and C'_1 simultaneously. Then B_1 and B'_1 can be added to $\mathsf{Tree}_{\mathsf{MC}}$ such that B_1 is the neighbour of B'_1, and depth of $\mathsf{Tree}_{\mathsf{MC}}$ grows by 1. Specifically, the adversary can broadcast C_1 to a set of miners, say S_1, and C'_1 to the remaining miners, say S'_1. Then miners in S_1 will accept chain C_1, while miners in S'_1 will accept C'_1. Let r_1 be the round at which B_1 and B'_1 are added to $\mathsf{Tree}_{\mathsf{MC}}$ and r_2 be the round at which the next block B_2 is mined. Notice that $r_2 - r_1$ is not influenced by the number of forks which the adversary generated at round r_1, and only the number of the rounds of delays affect the chain growth rate of $\mathsf{Tree}_{\mathsf{MC}}$.

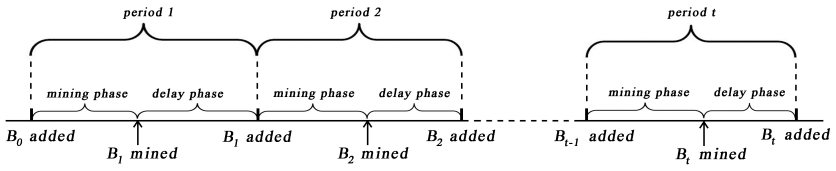

Fig. 1. The rounds during which t consecutive blocks are added to $\mathsf{Tree}_{\mathsf{MC}}$

Consider t consecutive blocks in $\mathsf{Tree}_{\mathsf{MC}}$ as shown in Figure 1. Block B_0 is added to the tree at round r_0 and B_t is added at round r_t. We divide those rounds from r_0 to r_t into t periods, and each period consists of the rounds during which the depth of $\mathsf{Tree}_{\mathsf{MC}}$ increases by 1.

Each period i consists of mining phase and delay phase. For each i, let B_i be the first block that mined in period i. The round at which block B_i is mined is

the end of mining phase. Let R^i_{mine} and R^i_{delay} denotes the number of rounds of mining phase and delay phase of period i, respectively. Let $R_{mine} = \Sigma^t_{i=1} R^i_{mine}$ and $R_{delay} = \Sigma^t_{i=1} R^i_{delay}$. So $R_{mine} + R_{delay} = r_t - r_0$.

Next, we show how to compute R_{mine} and R_{delay}. Let $f = 1-(1-p)^n$ be the probability that some miner succeeds in mining a block in a round. Since R^i_{mine}s are independent geometrically distributed variables such that $\Pr[R^i_{mine} = k] = (1-f)^{k-1}f$, the sum R_{mine} follows a negative binomial distribution $\mathsf{NB}(t, f)$. Due to Lemma 5 in Appendix A, we have

$$\Pr[R_{mine} \leq \frac{(1+\delta_1)t}{f}] \geq 1 - e^{-poly(\delta_1^2 t)}, \tag{11}$$

where $0 < \delta_1 < 1/2$.

In delay phase, if B_i is undelayable, it has to be added to $\mathsf{Tree}_{\mathsf{MC}}$ at the current round and $R^i_{delay} = 0$. Otherwise, the adversary can delay the chain for at most Δ rounds, $R^i_{delay} \leq \Delta$. It is obvious that $R_{delay} \leq t\Delta$. To get a lower upper bound, we need to consider the event that a undelayable block is mined during each delay phase. Indeed, if an undelayable block is mined within Δ rounds since the beginning of a delay phase, the adversary has to add such block to $\mathsf{Tree}_{\mathsf{MC}}$ and the delay phase is ended. Hence, the probability distribution of R^i_{delay} is defined as follows:

$$\Pr[R^i_{delay} = k] = \begin{cases} 1-\alpha, & \text{if } k = 0, \\ \alpha(1-(1-\alpha)f)^{k-1}(1-\alpha)f, & \text{if } 0 < k < \Delta, \\ \alpha(1-(1-\alpha)f)^{k}, & \text{if } k = \Delta, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

So we have

$$\begin{aligned} E[R^i_{delay}] &= \alpha(1-(1-\alpha)f)^{\Delta}\Delta + \sum_{k=1}^{\Delta-1} k\alpha(1-(1-\alpha)f)^{k-1}(1-\alpha)f \\ &= \frac{\alpha - \alpha\omega^{\Delta-1}[\omega + \Delta(1-\omega^2)]}{1-\omega}, \end{aligned}$$

where $\omega = 1-(1-\alpha)f$.

Since R^i_{delay}s are independent random variables with the same distribution, the expectation $E[R_{delay}] = \sum^t_{i=1} E[R^i_{delay}] = tE[R^i_{delay}]$. Using the Chernoff bound, we get

$$\Pr[R_{delay} < (1+\delta_2)tE[R^i_{delay}]] > 1 - e^{-\frac{\delta_2^2 tE[R^i_{delay}]}{3}}, \text{ for } 0 \leq \delta_2 \leq 1. \tag{13}$$

So on the condition that (11) and (13) hold, the chain growth rate of $\mathsf{Tree}_{\mathsf{MC}}$ is

$$g > \frac{t}{R_{mine} + R_{delay}} = \frac{t}{\frac{(1+\delta_1)t}{f} + (1+\delta_2)tE[R^i_{delay}]} = \frac{(1-\delta)f}{1 + fE[R^i_{delay}]}, \tag{14}$$

where δ is decided by picking sufficiently small δ_1 and δ_2.

Due to Lemma 3, the view of $(\Pi, \mathcal{C})$ has chain growth g with majority λ with probability at least $1 - 2e^{-poly(\kappa)}$ conditioned on that (11) and (13) hold. Therefore, given the view of $(\Pi, \mathcal{C})$, we have

$$\begin{aligned}&\Pr[\texttt{chain-increase}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r, r+T, \lambda) \geq gT] \\ &\geq 1 - 2e^{-poly(\kappa)} - e^{-poly(\delta_1^2 T)} - e^{-\frac{\delta_2^2 TE[R^i_{delay}]}{3}},\end{aligned}$$

which completes the proof of Theorem 1.

Remark. If $\alpha = 1$, then $E[R^i_{delay}] = \Delta$ and the chain growth rate is $\frac{(1-\delta)f}{1+f\Delta}$, which is the same as that of [22].

6.2 Common prefix

Theorem 2. *(Common prefix). Assume $0 < \alpha < 1 - np$ and $1/2 < \lambda \leq 1 - 8\alpha p\Delta$. The blockchain protocol $(\Pi, \mathcal{C})$ satisfies the common prefix property with parameter λ.*

Proof. Due to Lemma 4, it remains to prove that $\mathsf{Tree}_{\mathsf{MC}}$ have the common prefix property. Suppose the adversary's goal is to break the common prefix of $\mathsf{Tree}_{\mathsf{MC}}$ with depth $d+T$. That is, the adversary aims to make the length of the common prefix of all branches in $\mathsf{Tree}_{\mathsf{MC}}$ at most $d-1$.

Note that the depth of $\mathsf{Tree}_{\mathsf{MC}}$ can increase by 1 at most at each round due to Lemma 1. Therefore, in order to generate a fork in $\mathsf{Tree}_{\mathsf{MC}}$, the adversary has to broadcast more than one blocks in a round. If only one block is broadcasted, there will be only one branch in $\mathsf{Tree}_{\mathsf{MC}}$ according to Lemma 1 and the adversary fails to generate a fork.

In order to capture the attack for common prefix, we introduce the following game $\mathsf{Experiment}^{\mathsf{COMM}}_{A,(\Pi,\mathcal{C})}$, where the adversary generates a fork and tries to keep the branches of the fork as long as possible.

$\mathsf{Experiment}^{\mathsf{COMM}}_{A,(\Pi,\mathcal{C})}$: Run $(\Pi, \mathcal{C})$. Suppose that at current round r the depth of $\mathsf{Tree}_{\mathsf{MC}}$ is $d-1$ and there is no blocks being delayed and no forks in $\mathsf{Tree}_{\mathsf{MC}}$. Then the adversary A tries to generate a fork and extend the length of forks as follows.

1. Wait for new blocks to be mined. If the new block or blocks are mined at some round r' such that $r' > r$.
   - If more than one block are mined in the same round r', A broadcasts the corresponding chains and goes to step 3. That means a fork is generated and recorded in $\mathsf{Tree}_{\mathsf{MC}}$.
   - If only one block, say B, is mined,
     - If B is delayable, A delays the corresponding chain, say C_1, and goes to step 2;
     - Otherwise, go to step 1.

2. A tries to delay C_1 as long as possible. During these rounds of delays, A tries to generate a fork by "collecting" new blocks. If no block have been mined during these rounds, A fails to generate a fork and goes to step 1. Otherwise, go to step 3.
3. A tries to keep the fork of $\mathsf{Tree}_{\mathsf{MC}}$ as long as possible. If at least two branches of the fork are extended with T blocks, we say the adversary wins the common prefix game.

Since the adversary can always keep waiting and trying until a fork is created (in step 1 and step 2), the common prefix property is measured by the success probability of A in step 3.

Next, we consider a special event called **converge** which results in the failure of A. Suppose the depth of $\mathsf{Tree}_{\mathsf{MC}}$ increases to l at round r. Let B^* be the first block mined after round r and let r^* denote the round at which B^* is mined. The event **converge** satisfies the following conditions.

1. Only one miner succeeds in mining at round r^*.
2. The chain C^* which B^* lies in is undelayable, or C^* is delayable while there is no new block mined in following Δ rounds.

Note that if the event **converge** happens in step 3, then the depth of $\mathsf{Tree}_{\mathsf{MC}}$ increases by 1, e.g., from l to $l+1$.

When the depth of $\mathsf{Tree}_{\mathsf{MC}}$ increases to l at round r, the chains of all the miners are of length l. (Notice that the $(l+1)$th block can be mined only if a chain of length l is distributed). Then, if only one miner succeeds in r^* and generates an undelayable chain C^*, C^* will be the unique chain in $\mathsf{Tree}_{\mathsf{MC}}$ and A fails to extend the fork. If C^* is undelayable and there is no new block mined in following Δ rounds, A fails too.

Conditioned on that there exists some miner succeeding at round r^*, the probability of condition 1 is

$$\frac{np(1-p)^{n-1}}{1-(1-p)^n} > \frac{np(1-p)^{n-1}}{np} = (1-p)^{n-1} > 1-np \tag{15}$$

The probability of condition 2 is

$$1-\alpha+\alpha(1-p)^{n\Delta} > 1-\alpha+\alpha(1-np\Delta) = 1-\alpha np\Delta \tag{16}$$

Therefore,

$$Pr[\textbf{converge}] > (1-np)(1-\alpha np\Delta) > 1-np(1+\alpha\Delta) \tag{17}$$

The adversary can keep the fork for consecutive T blocks only if **converge** does not happen for consecutive T times, the probability of which is at most $(np(1+\alpha\Delta))^T$. So the probability that $\mathsf{Tree}_{\mathsf{MC}}$ has a common prefix with depth $d-T$ is at least $1-(np(1+\alpha\Delta))^T$.

If $\Delta \leq 1/np$, considering the assumption $\alpha < 1-np$, we have

$$np(1+\alpha\Delta) < np(1+\frac{1-np}{np}) = 1 \tag{18}$$

If $\Delta > 1/np$, the equality (16) can be replaced with $1-\alpha+\alpha(1-p)^{n\Delta} > 1-\alpha$, and the probability that $\mathsf{Tree}_{\mathsf{MC}}$ has a common prefix with depth $d-T$ is at least $1-(\alpha+np)^T$, where $\alpha + np < 1$.

To sum up, the probability that $\mathsf{Tree}_{\mathsf{MC}}$ has a common prefix path with depth $d-T$ is at least $1-negl(T)$, where $negl$ is a negligible function. Due to Lemma 4, the view of $(\Pi, \mathcal{C})$ satisfies $\texttt{common-prefix}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r, T, \lambda) = 1$ with probability at least $1 - 2e^{-poly(\kappa)}$. Therefore, given the view of $(\Pi, \mathcal{C})$, we have

$$\Pr[\texttt{common-prefix}^{(\Pi,\mathcal{C})}_{A,Z,\kappa}(r, T, \lambda) = 1] \geq 1 - 2e^{-poly(\kappa)} - negl(T),$$

which completes the proof of Theorem 2.

7 Long Delay Attack on Common Prefix

7.1 Long delay attack

Note that Theorem 2 is an asymptotic result, which means the common prefix property can hold when T is large enough. To illustrate the threat of long delay attack comprehensively, we present a concrete attack on the common prefix of $\mathsf{Tree}_{\mathsf{MC}}$ when Δ and α are "too" large relative to a fixed T.

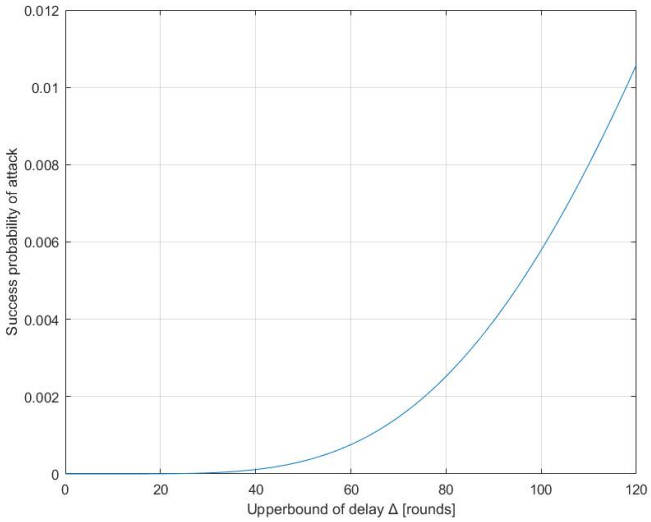

Fig. 2. For $\alpha = 0.8$ and $T = 6$, the success probability increases as Δ gets larger. In particular, the success probability grows much faster when $\Delta > 60$ (10 minutes). When $\Delta > 120$ (20 minutes), the success probability can reach about 1%.

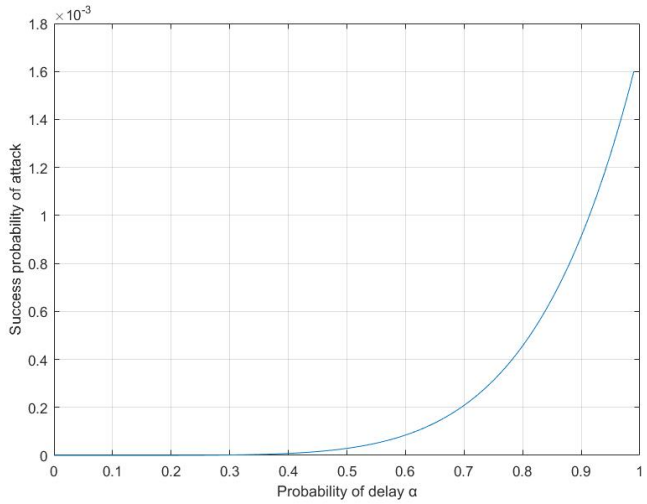

Fig. 3. For $\Delta = 60$ (10 minutes, the expected time of mining a block) and $T = 6$, the success probability increases as the probability of delay α get larger. As shown in the figure, the success probability increases much faster when $\alpha > 0.7$.

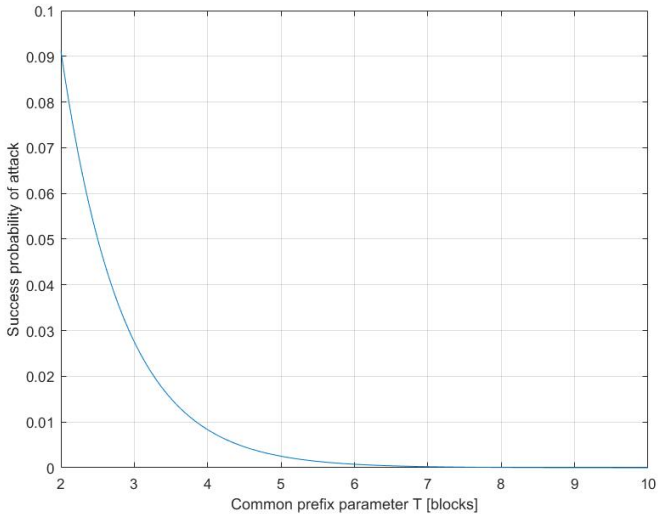

Fig. 4. For $\Delta = 60$ (10 minutes) and $\alpha = 0.8$, the success probability decreases as T gets larger. In particular, when $T \geq 6$, the success probability becomes extremely small.

Suppose that $\mathsf{Tree}_{\mathsf{MC}}$ has a fork with two branches[4]of depth 1, which lies in two chains, say chain A and chain B, respectively, and half of the miners accepted chain A and the other half accepted chain B. Then the adversary aims to increase the length of the two branches by T. Note that once the adversary need to broadcast two chains, he distributes in a way that the number of miners which accept one chain equals to that of miners which accepts the other chain. More details of the attack and related analysis are described in Appendix B. The success probability of such attack is

$$(\frac{f}{4} + (\alpha + \frac{f(1-2\alpha)}{4})\frac{f(1-p_{next}^{\Delta})}{2-2p_{next}})^{T} \tag{19}$$

where $p_{next} \approx \frac{(2-f(1-\alpha))(2-f)}{4}$.

For an experimental interpretation of the success probability of the attack, the parameters are set as follows: The time span of a round for full interaction is set to $10s$. Since the expected time to mine a block is about 10 minutes, the probability of all the miners succeeding in mining per round is about $f = 1/60$. Considering $n = 10^5$ miners in the network, we have $p \approx f/n \approx 1.67 \times 10^{-7}$. Let $\lambda = 99.8\%$, which satisfies the assumption $1/2 < \lambda \leq 1 - 8\alpha p\Delta$ if $\Delta < 1.5 \times 10^3$ (about 4.2 hours). In this case, the common prefix of $\mathsf{Tree}_{\mathsf{MC}}$ is the same as that of $(\Pi, \mathcal{C})$ with probability at least 99.95% due to Lemma 4.

Given the above parameters, Figure 2,3 and 4 reflect the success probability of long delay attack when Δ, α and T varies. As shown in those figures, the adversary without any hash power may threaten the common prefix property of blockchain protocol especially when Δ and α are too large relative to the fixed T.

7.2 Balance attack

Our attack is reminiscent of the balance attacks introduced by [20], since both attacks can create or maintain forks by splitting honest miners into subgroups of similar mining power. Main differences between the original balance attack in [20] and ours are as follows.

- The goal of the original balance attack is to make the target branch selected as the main chain, while the goal of ours is to maintain the forks for as long as possible.
- The attacker in the original balance attack requires a fraction, say 20%, of mining power to launch attack, while our attack as well as N-confirmation double spending attack in [14] does not require any mining power.
- The original balance attack disrupts the communication between subgroups by delaying messages and those isolated subgroups mine their own blockchains independently. Our attack delays the new block (or blockchain) as soon as it is successfully mined, e.g., the attacker "eclipses" the miner which mines

[4] Here the branch starts from the block where the fork begins and ends with the last block.

a new block. Then the attacker delivers different blockchains to different subgroups once he obtains enough blockchains.

According to Theorem 5 of [20], we can evaluate the effectiveness of balance attack on bitcoin protocol. Table 1 shows the time of delays (in minutes) required by the original balance attack and ours, where we only consider the ideal case for the attacker of balance attack. More precisely, we assume that all the blocks mined in balance attack can be added to the main chain. For more details of balance attack, we refer to [20].

Table 1. Delays for balance attack and our long delay attack (minutes). $f = 1/60$ and $T = 6$. ϵ denotes the success probability of the attack. ρ denotes the fraction of mining power owned by the adversary in balance attack. α denotes the probability of delay in our attack. "-" denotes that the corresponding success probability cannot be achieved. For example, the maximum success probability of our attack is about 0.55 when $\alpha = 0.95$ and hence cannot reach 0.9.

Types of attack		Success probability		
		$\epsilon = 0.1$	$\epsilon = 0.5$	$\epsilon = 0.9$
Balance attack	$\rho = 0.1$	8055	11230	11920
	$\rho = 0.2$	1790	2495	4426
	$\rho = 0.3$	696	970	1724
Our attack	$\alpha = 0.85$	43.6	-	-
	$\alpha = 0.95$	26.6	78.4	-
	$\alpha = 1$	22.9	44.2	80.8

Although Table 1 shows that the balance attack requires longer delays than ours, we emphasize that it is not fair to say which attack is better. First, the goals are different. Second, the balance attack only considers the case that the attacker can always delay the message successfully, while our attack considers different probability of delay. Besides, the success probability estimation of balance attack on bitcoin, which is obtained by applying the result on GHOST [20] directly, is not tight and can be further improved.

Acknowledgements. We would like to thank the anonymous reviewers of ASIACRYPT 2018 for their insightful and helpful comments. We are also grateful to Siu Ming Yiu, Zhengyu Zhang, Yingnan Deng, Shichen Wu and Xianrui Qin for interesting discussions. Puwen Wei and Quan Yuan were supported by the National Natural Science Foundation of China (No. 61502276 and No. 61572293). Puwen Wei was also supported by the Chinese Major Program of National Cryptography Development Foundation (No. MMJJ2017012) and the Fundamental Research Funds of Shandong University (No. 2016JC029).

References

1. Badertscher, C., Garay, J., Maurer, U., Tschudi, D., Zikas, V.: But why does it work? a rational protocol design treatment of bitcoin. In: Nielsen, J., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 34–65. Springer, Cham (2018)
2. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payment from bitcoin. IEEE Symposium on Security and Privacy pp. 459–474 (2014)
3. Carlsten, M., Kalodner, H.A., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: ACM CCS 2016. pp. 154–167. ACM Press, New York (2016)
4. Daian, P., Pass, R., Shi, E.: Snow white: Provably secure proofs of stake. IACR Cryptology ePrint Archive, Report 2016/919 (2016)
5. David, B., Gaži, P., Kiayias, A., Russell, A.: Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol. In: Nielsen, J., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10821, pp. 66–98. Springer, Cham (2018)
6. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: 13th IEEE International Conference on Peer-to-Peer Computing. pp. 1–10. IEEE Computer Society Press (2013)
7. Dubhashi, D.P., Panconesi, A.: Concentration of measure for the analysis of randomized algorithms. Cambridge University Press (2009)
8. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 436–454. Springer, Berlin, Heidelberg (2014)
9. Eyal, I., Sirer, E.G.: The miner's dilemma. In: 2015 IEEE Symposium on Security and Privacy. vol. 2015-7, pp. 89–103. IEEE Computer Society Press (2015)
10. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Berlin, Heidelberg (2015)
11. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 291–323. Springer, Cham (2017)
12. Gervais, A., Karame, G.O., Wust, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: ACM CCS 2016. pp. 3–16. ACM Press (2016)
13. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. IACR Cryptology ePrint Archive, Report 2017/454 (2017)
14. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoins peer-to-peer network. In: Jung, J. (ed.) 24th USENIX Security Symposium. pp. 129–144. USENIX Association (2015)
15. Kiayias, A., Koutsoupias, E., Kyropoulou, M., Tselekounis, Y.: Blockchain mining games. In: 2016 ACM Conference on Economics and Computation. pp. 365–382. ACM Press (2016)
16. Kiayias, A., Panagiotakos, G.: Speed-security tradeoffs in blockchain protocols. IACR Cryptology ePrint Archive: Report 2015/1019 (2016)
17. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017)

18. Miller, A., LaViola, J.J.: Anonymous byzantine consensus from moderately-hard puzzles: A model of bitcoin. University of Central Florida. Tech Report, CS-TR-14-01 (2014)
19. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
20. Natoli, C., Gramoli, V.: The balance attack against proof-of-work blockchains: The R3 testbed as an example. Computing Research Repository (2016), arXiv:1612.09426
21. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In: 2016 IEEE European Symposium on Security and Privacy. vol. 142, pp. 305–320. IEEE Computer Society Press (2016)
22. Pass, R., Seeman, L., abhi shelat: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J., Nielsen, J. (eds.) Advances in Cryptology - EUROCRYPT 2017. LNCS, vol. 10211, pp. 643–673. Springer-Verlag, Cham (2017)
23. Pass, R., Shi, E.: Fruitchains: A fair blockchain. In: ACM Symposium on Principles of Distributed Computing. pp. 315–324. ACM Press (2017)
24. Pass, R., Shi, E.: The sleepy model of consensus. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 380–409. Springer, Cham (2017)
25. Pass, R., Shi, E.: Thunderella: Blockchains with optimistic instant confirmation. In: Nielsen., J., Rijmen, V. (eds.) EUROCRYPT 2018. vol. 10821, pp. 3–33. Springer (2018)
26. Rosenfeld, M.: Analysis of bitcoin pooled mining reward systems. arXiv preprint:1112.4980 (2011), arXiv:1112.4980
27. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 515–532. Springer, Berlin, Heidelberg (2016)
28. Schrijvers, O., Bonneau, J., Boneh, D., Roughgarden, T.: Incentive compatibility of bitcoin mining pool reward functions. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 477–498. Springer, Berlin, Heidelberg (2016)
29. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. IACR Cryptology ePrint Archive: Report 2013/881 (2017)
30. Teutsch, J., Jain, S., Saxena, P.: When cryptocurrencies mine their own business. In: Grossklags, J., Preneel, B. (eds.) FC 2016. LNCS, vol. 9603, pp. 499–514. Springer, Berlin, Heidelberg (2016)
31. Zohar, A.: Bitcoin: under the hood. In: Communications of the ACM. vol. 58, pp. 104–113. ACM Press (2015)

A Chernoff bound for negative binomial distribution

Lemma 5. *Let $X_1, X_2, ..., X_k$ be independent random variables, such that for all $i \in [k]$ and integer $m \geq 1$, $Pr[X_i = m] = (1-p)^{m-1}p$. Let $X = \Sigma_{i=1}^{k} X_i$, the variable X is said to have a negative binomial distribution $\mathsf{NB}(k,p)$, and for $\delta \in (0, \frac{1}{2})$*

$$\Pr[X \leq (1-\delta)\frac{k}{p}] < e^{-poly(\delta^2 k)} \tag{20}$$

$$\Pr[X \geq (1+\delta)\frac{k}{p}] < e^{-poly(\delta^2 k)} \tag{21}$$

Proof. Let $t = \frac{(1-\delta)k}{p}$ and $\epsilon = \frac{1}{1-\delta} - 1 \in (0,1)$. Here, $k = \frac{pt}{1-\delta} = (1+\epsilon)pt$. Let $Y_1, Y_2, ..., Y_{\lfloor t \rfloor}$ be independent random boolean variables, such that for all $i \in \{1, \ldots, \lfloor t \rfloor\}$, $\Pr[Y_i = 1] = p$. $Y = \Sigma_{i=1}^{\lfloor t \rfloor} Y_i$. Due to the Chernoff bound [7], we have

$$\Pr[Y \geq k] \leq \Pr[Y \geq (1+\epsilon)p\lfloor t \rfloor] < e^{-\frac{\epsilon^2 p \lfloor t \rfloor}{3}} \tag{22}$$

Since

$$p\lfloor t \rfloor = pt - p(t - \lfloor t \rfloor) > pt - p = (1-\delta)k - p \tag{23}$$

we have

$$\Pr[Y \geq k] < e^{-\frac{\epsilon^2}{3}((1-\delta)k-p)} < e^{-\frac{\delta^2 k}{3(1+\delta)} + \frac{\epsilon^2 p}{3}} < e^{-\frac{\delta^2 k}{3} + \frac{p}{3}} \tag{24}$$

Consider the event $Y \geq k$. If it happens, there are at least k successes in $\lfloor t \rfloor$ Bernoulli trials. In other words, it takes us at most $\lfloor t \rfloor$ experiments to achieve the kth successes. X_i is considered as the number of Bernoulli trials needed to get one success. So the event $Y \geq k$ is equivalent to the event $X \leq \lfloor t \rfloor$. Hence,

$$\Pr[X \leq t] = \Pr[X \leq \lfloor t \rfloor] = \Pr[Y \geq k] < e^{-\frac{\delta^2 k}{3} + \frac{p}{3}} \tag{25}$$

That completes the proof of inequality (20). Similarly, inequality (21) can be proved if $t = \frac{(1+\delta)k}{p}$ and $\epsilon = 1 - \frac{1}{1+\delta}$.

B Long Delay Attack on Common Prefix

Suppose chain A and chain B are in $\mathsf{Tree}_{\mathsf{MC}}$, such that chain A is similar to chain B except that only the last blocks are different. For convenience, let G_A and G_B denote the set of miners which accept chain A and chain B, respectively. The number of miners in group A equals to that of group B, i.e., $|G_A| = |G_B| = n/2$. Note that G_A or G_B is not fixed and will be changed due to the adversary delivery. Then the adversary waits for a new block. We say round r is successful if there is a new block mined at round r. Let $\gamma(n,p) = 1-(1-p)^n$. If p is small, we have $\gamma(n,p) \approx np$. Obviously, the probability that a round is successful is $f = \gamma(n,p)$. When a successful round appears, consider the following cases:

1. There is at least one block mined in each of the two branches. That means, chain A and chain B are extended at the same round. The adversary distributes the two chains in a way that the number of miners accepting chain A and the number of miners accepting chain B are equal. As a result, the adversary succeeds in extending the length of the fork by 1. Since the probability of mining a block in one chain is $\gamma(\frac{n}{2}, p)$, the probability that this case happens is
$$\frac{\gamma(\frac{n}{2},p) \cdot \gamma(\frac{n}{2},p)}{\gamma(n,p)} \approx \frac{\frac{1}{4}n^2p^2}{np} = \frac{np}{4} \tag{26}$$
2. There is at least one *undelayable* block mined in only one of the branches (without loss of generality, chain A) while no block mined in the another chain (chain B). In this case, the new chain A is broadcasted by the adversary,

while the length of chain B remains the same. So the useless blocks in chain B is deleted due to **DeleteBlock** and the adversary fails to extend the fork. The probability of this case is

$$\frac{2\gamma(\frac{n}{2},(1-\alpha)p)(1-\gamma(\frac{n}{2},p))}{\gamma(n,p)} \approx \frac{(1-\alpha)np(1-\frac{np}{2})}{np} = \frac{(1-\alpha)(2-np)}{2} \tag{27}$$

3. Otherwise, all the blocks mined at this round are delayable and in only one branch (without loss of generality, chain A). That means, the adversary neither succeeds nor fails at this round. The adversary can delay the new chain A and keep waiting for a new block in chain B in the following Δ rounds. Due to equations (26)(27), the probability that the adversary needs to delay the chain is

$$1 - \frac{np}{4} - \frac{(1-\alpha)(2-np)}{2} = \alpha + \frac{np(1-2\alpha)}{4} \tag{28}$$

Then, the adversary keeps the chain A delayed and waits for a new block to be mined in chain B. At each of the following rounds, there are three cases to be discussed:

(a) If G_B succeeds in mining a block, the chain B can be extended and adversary distributes the delayed chain A and the new chain B in a way that $|G_A| = |G_B|$. As a result, the adversary succeeds in extending the length of chain A and chain B by 1. The probability of this case is $\gamma(\frac{n}{2},p) \approx \frac{np}{2}$.

(b) If G_B does not succeed in mining a block while G_A mines an *undelayable* block. Then, the new chain A should be distributed, which means only chain A in $\mathsf{Tree}_{\mathsf{MC}}$ is extended. So the adversary fails.

(c) If G_A does not mine an undelayable block while G_B does not succeed in mining a block. The adversary checks whether the number of rounds for A being delayed exceeds Δ. If it exceeds Δ, the adversary has to broadcast chain A and fails. Otherwise, the adversary keeps chain A delayed and goes to the next round, where the probability of this event is

$$p_{next} = (1-\gamma(\frac{n}{2},(1-\alpha)p)) \cdot (1-\gamma(\frac{n}{2},p)) \approx \frac{(2-np(1-\alpha))(2-np)}{4}. \tag{29}$$

In a word, in case 3, the adversary can succeed with probability $\frac{np}{2}$ per round, or can go to the next round with probability p_{next}. Conditioned on case 3 happens at round r, the probability for the adversary succeeding at round $r+1$ is $\frac{np}{2}$ and the probability of success at round $r+2$ is $p_{next} \cdot \frac{np}{2}$. Similarly, conditioned on case 3 happens at round r, the probability of success at round $r+i$ is ${p_{next}}^{i-1} \cdot \frac{np}{2}$. Since the adversary only has Δ rounds for trying, the probability for adversary to succeed during those Δ rounds in case 3 is

$$\sum_{i=1}^{\Delta} {p_{next}}^{i-1} \cdot \frac{np}{2} = \frac{np(1-p_{next}^{\Delta})}{2-2p_{next}}. \tag{30}$$

Considering case 1, 2 and 3, the probability of the adversary succeeding in increasing the length of branches by 1 for a successful round is

$$\frac{np}{4} + (\alpha + \frac{np(1-2\alpha)}{4})\frac{np(1-p_{next}^{\Delta})}{2-2p_{next}}. \tag{31}$$

Then the adversary waits for another successful round and executes as described above.

We say the adversary's long delay attack is successful, if the adversary succeeds in increasing the length of the fork by 1 for consecutive T times. Therefore, the success probability of our long delay attack is

$$(\frac{f}{4} + (\alpha + \frac{f(1-2\alpha)}{4})\frac{f(1-p_{next}^{\Delta})}{2-2p_{next}})^{T}. \tag{32}$$

where $np \approx f$.