FruitChains: A Fair Blockchain

Rafael Pass*
Cornell Tech
rafael@cs.cornell.edu

Elaine Shi†
Cornell University
elaine@cs.cornell.edu

September 21, 2016

Abstract

Nakamoto's famous *blockchain* protocol enables achieving consensus in a so-called *permissionless setting*—anyone can join (or leave) the protocol execution, and the protocol instructions do not depend on the identities of the players. His ingenious protocol prevents "sybil attacks" (where an adversary spawns any number of new players) by relying on *computational puzzles* (a.k.a. "moderately hard functions") introduced by Dwork and Naor (Crypto'92). Recent work by Garay et al (EuroCrypt'15) and Pass et al (manuscript, 2016) demonstrate that this protocol provably achieves *consistency* and *liveness* assuming a) honest players control a majority of the computational power in the network, b) the puzzle-hardness is appropriately set as a function of the maximum network delay and the total computational power of the network, and c) the computational puzzle is modeled as a random oracle.

Assuming honest participation, however, is a strong assumption, especially in a setting where honest players are expected to perform a lot of work (to solve the computational puzzles). In Nakamoto's Bitcoin application of the blockchain protocol, players are *incentivized* to solve these puzzles by receiving rewards for every "blocks" (of transactions) they contribute to the blockchain. An elegant work by Eyal and Sirer (FinancialCrypt'14), strengthening and formalizing an earlier attack discussed on the Bitcoin forum, demonstrates that a *coalition* controlling even a minority fraction of the computational power in the network can gain (close to) 2 times its "fair share" of the rewards (and transation fees) by deviating from the protocol instructions. In contrast, in a *fair* protocol, one would expect that players controlling a ϕ fraction of the computational resources to reap a ϕ fraction of the rewards.

In this work, we present a new blockchain protocol—the *FruitChain* protocol—which satisfies the same consistency and liveness properties as Nakamoto's protocol (assuming an honest majority of the computing power), and additionally is δ*-approximately fair*: with overwhelming probability, any honest set of players controlling a ϕ fraction of computational power is guaranteed to get at least a fraction $(1-\delta)\phi$ of the blocks (and thus rewards) in *any* $\Omega(\frac{\kappa}{\delta})$ length segment of the chain (where κ is the security parameter).

As a consequence, if this blockchain protocol is used as the ledger underlying a cryptocurrency system, where rewards and transaction fees are *evenly distributed* among the miners of blocks in a length κ segment of the chain, no coalition controlling less than a majority of the computing power can gain more than a factor $(1+3\delta)$ by deviating from the protocol (i.e., honest participation is an $\frac{n}{2}$*-coalition-safe* 3δ*-Nash equilibrium*).

Finally, the fruit chain protocol enables decreasing the *variance* of mining rewards and as such significantly lessens (or even obliterates) the need for mining pools.

*Supported in part by NSF Award CNS-1217821, NSF Award CNS-1561209, AFOSR Award FA9550-15-1-0262, a Microsoft Faculty Fellowship, and a Google Faculty Research Award.

†Supported in part by NSF Award CNS-1314857, CNS-1514261, CNS-1544613, CNS-1561209, CNS-1601879, CNS-1617676, an Office of Naval Research Young Investigator Program Award, a Packard Fellowship, a Sloan Fellowship, Google Faculty Research Awards, and a VMWare Research Award.

1 Introduction

Distributed systems have been historically analyzed in a *closed* setting—a.k.a. the *permissioned setting*—in which the number of participants in the system, as well as their identities, are common knowledge. In 2008, Nakamoto [Nak08] proposed his celebrated "blockchain protocol" which attempts to achieve consensus in a *permissionless* setting: anyone can join (or leave) the protocol execution (without getting permission from a centralized or distributed authority), and the protocol instructions do not depend on the identities of the players. The core blockchain protocol (a.k.a. "Nakamoto consensus", or the "Bare-bones blockchain protocol"), roughly speaking, is a method for maintaining a *public*, *immutable* and *ordered* ledger of records (for instance, in the Bitcoin application, these records are simply transactions); that is, records can be added to the *end* of the ledger at any time (but only to the end of it); additionally, we are guaranteed that records previously added cannot be removed or reordered and that all honest users have a *consistent view* of the ledger—we refer to this as *consistency.*

The problem with the permissionless setting is that an attacker can trivially mount a so-called "sybil attack"—it simply spawns lots of players (that it controls) and can thus easily ensure that it controls a majority of all the players. Indeed, Barak et al [BCL+05] proved that this is a fundemental problem with the permissionless model. Nakamoto blockchain protocol overcomes this issue by relying on "computational puzzles"—a.k.a. *moderately hard functions* or *proofs of work*—put forth by Dwork and Naor [DN92]: rather than attempting to provide robustness whenever the majority of the participants are honest (since participants can be easily spawned in the permissionless setting), it attempts to provide robustness as long as a *majority of the computing power* is held by honest participants.

Nakamoto's protocol in a nutshell Roughly speaking, players "confirm" records/transactions by "mining blocks of transactions" through solving some computational puzzle that is a function of the transactions and the history so far. More precisely, each participant maintains its own local "chain" of "blocks" of records/messages—called the *blockchain.* Each block consist of a triple $(h_{-1}, \eta, \mathsf{m})$ where h_{-1} is a pointer to the previous block in chain, m is the record component of the block, and η is a "proof-of-work"—a solution to a computational puzzle that is derived from the pair (h_{-1}, m). The proof of work can be thought of as a "key-less digital signature" on the whole blockchain up until this point.

Concretely, Nakamoto's protocol is parametrized by a parameter p—which we refer to as the *mining hardness parameter*, and a proof-of-work is deemed valid if η is a string such that $\mathsf{H}(h_{-1}, \eta, \mathsf{m}) < D_p$, where H is a hash function (modeled as a random oracle) and D_p is set so that the probability that an input satisfies the relation is less than p. At any point of the protocol execution, each participant attempts to increase the length of its own chain by "mining" for a new block: upon receiving some record m, it picks a random η and checks whether η is a valid proof of work w.r.t. m and h_{-1}, where h_{-1} is a pointer to the last block of its current chain; if so, it extends is own local chain and broadcast it to the all the other participants. Whenever a participant receives a chain that is longer than its own local chain, it replaces its own chain with the longer one.

Security of Nakamoto's protocol assuming Honest Majority of Computing Power Several recent work have analyzed the blockchain protocol under the assumption that a majority of the computational power is held by honest players (and assuming that the computational puzzled is modeled as a random oracle):

- Nakamoto [Nak08] proves that a certain natural attack does not work to break consistency;
- Garay, Kiayas and Leonardos [GKL15] prove that consistency holds assuming the adversary does not delay messages (i.e., in a synchronous model);

- Sompolinsky and Zohar [SZ15] show that so-called "non block-witholding" attacks do not work to break consistency, even in the presence of bounded network delays;
- Pass, Seeman and Shelat [PSS16], finally, demonstrate consistency w.r.t. *all* attacks, assuming bounded (but adversarial) network delays (as long as the mining hardness is appropriately set as a function of the maximum network delay).[1]

Following [GKL15, KP15], [PSS16] also defines an *abstract* notion of a blockchain and identifies three desirable security properties such an abstraction should satisfy: roughly speaking, those are *consistency*—that honest users agree on the chain; *chain quality*—that the chain contains a sufficient fraction of blocks contributed by honest players, we return to this property shortly; and *chain growth*—that the blockchain grows at some positive speed and thus we can ensure new transactions can be added. As shown in [PSS16], these properties, in particular, imply that a blockchain can be used to obtain an "immutable public ledger" [GKL15] (as described above) which guarantees standard properties of "consitency" and "liveness"; further applications of those properties can be found in [PS16].

Selfish mining and Fairness (chain quality) The above analyses, however, all assume that a majority of the computing power is controlled by honest players, and that honest players correctly execute the protocol. Assuming such honest participation is a strong assumption, especially in a setting where honest players are expected to perform a lot of work (to solve the computational puzzles)—why would we expect players to want to participate if it is costly! (This can be formalized in the Game-Theory with Costly computation framework of Halpern and Pass [HP15]). In Nakamoto's ingenious Bitcoin application of the blockchain protocol, players are thus *incentivized* to solve these puzzles by receiving, so-called, *mining rewards* for every "blocks" (of transactions) they contribute to the blockchain; additionally, the miners also receive *transaction fees* for all the transactions that are confirmed in the block.

However, as pointed out already in a discussion on the Bitcoin forum [mtg10] in 2010, a *coalition* controlling even a minority fraction $\rho < 1/2$ of the computational power in the network can contribute a significantly larger fraction of blocks than its "fair share" ρ (and thus reap more than its fair rewards) assuming the adversary controls the delivery of messages on the network: Whenever the adversarial coalition mines a new block, it simply *withholds it* (not sharing it with the honest players), and only releases it when some honest player mines a new block—if the adversary controls the network it can ensure that all honest players receive the adversarial block before the block mined by the honest players, and as such, it effectively "erases" the honest player's block replacing it with its own block. Thus, by the time T blocks have been mined, in expectation ρT of them are adversarial, and $(1-\rho)T$ of them are "honest", but since the adversary is able to use each one of its blocks to "replace" an honest block, the chain only grows by $(1-\rho)T$ blocks and thus the expected fraction of adversarial blocks becomes $\frac{\rho}{1-\rho}$. Thus, if the adversary controls close to a fraction $\frac{1}{2}$ of the computational resources, it will get almost all of the blocks, and thus almost *twice* its fair share of the rewards.[2] A surprising and beautiful work by Eyal and Sirer [ES14] presents an even stronger attack, which they also analytically study, showing that a more sophisticated selfish mining strategy can be successful even without controlling the network traffic! An even sharper attack is presented and analyzed by Sapirshtein, Sampolinsky, Zohar [SSZ16].

Following Garay et al [GKL15], we refer to a high-probability *lowerbound* on the fraction of blocks "contributed" by honest players in *any sufficiently long window* chain, as the *chain quality*

[1]Additionally, the protocol is shown to be insecure if the mining hardness is too small.

[2]The discussion on the Bitcoin forum did not contain an an analysis of the attack, but rather just simulation results demonstrating the power of it.

of the blockchain protocol.[3] Ideally, one would have hoped for a chain quality of

$$1-\rho$$

but the above attacks show that the chain quality can be signifcantly worse; on the positive side, Garay et al [GKL15] (in synchronous networks) and Pass et al [PSS16] (also in networks with adversarial bounded delays) show that Nakamoto's protocol achieves chain quality close to

$$1-\frac{\rho}{1-\rho}$$

when the mining hardness parameter is appropiately set, and thus the above-mentioned block withholding attack is optimal.

But, as mentioned, when ρ approaches $\frac{1}{2}$ the chain quality also approaches 0—that is, the adversary gets *all* the blocks (and thus rewards and fees) although it only controls half the computational resources, and consequently honest players putting in a lot of work get nothing! (And an adversary controlling only $\frac{1}{3}$ of the resources may get half of the rewards.)

Ideally, we would want a block chain protocol that is *fair*: namely, honest players contributing a $\phi \leq 1-\rho$ fraction of the computational resources get a ϕ fraction of the blocks (and thus rewards) in any sufficiently long window—in particular, this implies that a chain quality of $1-\rho$. The construction of such a blockchain protocol has remained open:

Does there exists a **fair** *blockchain protocol in the permissionless setting?*

1.1 Our Results

In this work we address this question. We present a blockchain protocol satisfying "close-to-optimal" fairness: We say that a blockchain protocol is δ*-approximately fair* w.r.t. ρ attackers if, with overwhelming probability, any $\phi \leq 1-\rho$ fraction coalition of *honest* users is guaranteed to get at least a $(1-\delta)\phi$ fraction of the blocks in every sufficiently long window, even in the presence of an adversary controlling up to a ρ fraction of the computing power. Note that this condition trivially implies that the protocol satisfies $(1-\delta)(1-\rho)$ chain quality, by considering the full set of honest users (i.e., $\phi = 1-\rho$). When $\rho \leq \frac{1}{2}$, $(1-\delta)(1-\rho) \geq 1-(1+\delta)\rho$[4], and thus as a consequence we get than the adversary coaltion cannot get more than $(1+\delta)$ times its "fair" share.

Theorem 1.1 (Informally stated). *Assume $\rho < \frac{1}{2}$. Then, for every $\delta > 0$, there exists a blockchain protocol that satisfies consistency, growth and δ-approximate fairness (and thus also chain quality $(1-\delta)(1-\rho) \geq 1-(1+\delta)\rho$).*

From Fairness to Incentive Compatibility We remark that any secure blockchain protocol that satisfies δ-approximate fairness (where $\delta < 0.3$) w.r.t $T(\kappa)$ length windows can be used as the ledger underlying a cryptocurrency system while ensuring 3δ*-incentive compatibility* if players (i.e. miners) only care about how much money they receive—that is, a miner's utility is the sum of the rewards and transaction fees it receives (potentially times some constant).[5]

Consider a crypto-currency which uses a blockchain protocol as the underlying ledger; we omit a formalization of what this means, but have in mind a system such as Bitcoin where rewards and

[3] The fact that we require this property to hold for *any* window (as opposed to just a notion of quality in expectation) is important for application. We return to this point below.

[4] When $\rho \leq \frac{1}{2}$, $(1-\delta)(1-\rho) = 1-\delta-\rho+\delta\rho = 1-[\delta+(1-\delta)\rho] \geq 1-[2\delta\rho+(1-\delta)\rho] = 1-(1+\delta)\rho$

[5] This may not always be a realistic assumption. For instance, a miner can care about what transactions get added into the blockchain etc, but following earlier approaches to modeling incentives in blockchains (e.g., [ES14]), we focus only on miners' monetary rewards.

transaction fees are somehow distributed among the miners of blocks—for instance, recall that in Bitcoin, the miner of a block receives a mining reward as well as all the transaction fees contained in the block it mined.

We say that *honest mining is a ρ-coalition-safe ϵ-Nash equilibrium* if, with overwhelming probability, no $\rho' < \rho$ fraction coalition can gain more than a multiplicative factor $(1 + \epsilon)$ in utility, no matter what transactions are being processed—formally, consider some environment providing transactions into the system. We restrict to a setting where the *total* rewards and transaction fees during the run of the system is some fixed constant V: [6]

We now remark that if rewards and transaction fees are *evenly distributed* among the (miners of the) blocks in the $T(\kappa)$-length segment of the chain following the block,[7] then it follows that honest mining is a ρ-coalition-safe 3δ-Nash equilibrium as long as the underlying blockchain satisfies δ-approximate fairness w.r.t. ρ attackers: as noted above, fairness implies that no matter what deviation the coalition performs, with overwhelming probability, the fraction of adversarial blocks in any $T(\kappa)$-length window of the chain is upperbounded by $(1 + \delta)\rho$ and thus the total amount of compensation received by the attacker is bounded by $(1 + \delta)\rho \cdot V$; in contrast, by fairness, if the coalition had been following the honest protocol, they are guaranteed to receive at least $(1 - \delta)\rho \cdot V$; thus, the multiplicative increase in utility is

$$\frac{1+\delta}{1-\delta} \leq 1 + 3\delta$$

when $\delta < 0.3$.[8]

To see why the "standard" bitcoin approach of giving all rewards and fees to the miner of the block does not work, consider an freshly mined (honest) block containing a transaction with a very high transaction fee. A coalition controlling a constant fraction of the computing power would have a huge incentive to "drop" this block and instead try to mine a new block which contains it. Fairness does not prevent such an attack, and indeed, even in our protocol such an attack will be successful with constant probability. (Indeed, it has been informally conjectured in the bitcoin community that ϵ-incentive compatibility is impossible to achieve in the presence of transaction fees, due to exactly this reason. Our method of distributing the fees over a segment overcomes this "barrier".)

Fairness v.s. "Eventual Fairness" A diffrent notion of fairness, which we here refer to as "eventual fairness", was informally considered in an elegant work of Lewenberg, Sampolinsky and Zohar [LSZ15]: roughly speaking, rather than requiring fairness in *any (sufficiently long) window*, their notion only requires fairness *in expectation over an infinitely long run of the system.* To better understand the difference between these notions, consider the *inclusive blockchain* protocol considered in [LSZ15]: Roughly speaking, it is a variant of Nakamoto's blockchain where "forked" blocks can be picked up and put back into the main chain (as records). The idea is that if an attacker starts dropping honest players' blocks (as in the selfish mining attack), they can be picked up by subsequent honest miners and included into the chain (looking forward, we will rely on a similar intuition). Note, however, that an attacker can now secretly mine blocks on a fork and then at any point in the execution have those blocks included in the main chain, thereby creating an arbitrarily long sequence of "malicious" blocks and thus "poisoning" a large segement of the chain.

[6]The analysis directly extends to a setting where the total rewards and fees are only guaranteed to be withing some multiplicative factor $(1 + \delta')$ of V at the cost of a degradation of the quality of the Nash equilibrium (i.e., increasing the ϵ).

[7]We could have equally picked a segment of the chain preceeding the block, but that would lead to complications in the startup phase when the chain is too short

[8]Let us remark that an alternative approach would be to give the whole mining reward to the miner of a block (as in Bitcoin) but still distribute the transaction fees among the group of miners in a $T(\kappa)$-segment of the chain. This approach works by the same analysis as long as mining rewards are *fixed* throughout the experiment (which is not the case for e.g., Bitcoin where mining rewards decrease over time).

Such a notion of eventual fairness can be used to guarantee that if mining rewards are kept constant, in the limit, the expected amount of mining rewards received by an attacker is proportional to its computational power. However, it is not clear how to (even in the limit) deal with mining rewards whose amount change over time, and perhaps more importantly, how to deal with transaction fees (which may differ significantly between transactions). (Finally, [LSZ15] only analyze their protocol w.r.t. to some specific attack strategies, and thus even obtaining a blockchain protocol which provably satisfies such an eventual notion of fairness was open.)

1.2 Proof Overview

Roughly speaking, our protocol, which we refer to as the *FruitChain* protocol, will be running an instance of Nakamoto's blockchain protocol, but instead of directly storing the records m inside the blockchain, the records are put inside "fruit" f; these fruit themselves requires solving some proof of work, with a *differerent hardness parameter* p_f; additionally, we require the fruits to "hang" from a block in the chain which is not too "far" from the block which records the fruit—more specifically, the fruit needs to "point" to an earlier block in the chain which is not too far from the block containing it (and thus, the fruit could not have been mined "too" long ago)—we refer to such a fruit as being *recent*. In this new protocol, the fruits play the roles of "blocks"—i.e., *"orange is the new block"*[9]—and chain quality is thus defined in terms of fruits.

In each round, honest players simultaneously mine for a fruit and a block (for Nakamoto's blockchain) by making one invokation of the hashfunction—this follows the 2-for-1 trick of [GKL15] where, say, the prefix of the output of H determines whether fruit mining is successful, and the suffix is used to determine whether block mining is successful. Whenever a player successfully mines a fruit it broadcasts it to all other players; fruits that have not yet been recorded in the blockchain (and that are still recent) are stored in a buffer and all honest players next attempt to add them to the blockchain.

Intuitively, the reason why "selfish mining" fails is that even if an adversary tries to "erase" some block mined by an honest player (which contains some honest fruits), by the chain growth and chain quality properties of the underlying blockchain, eventually an honest player will mine a new block which is stable and this honest player will include the fruit in it—in fact, the time before such an "honest block" arrives is short enough for the fruit to still be "recent" at the time of the honest block arriving.

Intuitively, the reason why we require fruits to be recent is to prevent a different kind of attack: without it, an attacker could *withhold fruits*, and suddenly release lots of them at the same time, thereby creating an very high fraction of adversarial fruit in some segment of the (fruit) chain. By requiring the fruits to be recent, we prevent the adversary from squirreling away (too many of) its fruits: since the underlying blockchain has a guaranteed chain growth, we can upperbound the extra amount of time the attacker can withold fruits and thus upperbound the number of extra fruits it can release in any window.

Related Techniques As already mentioned, the work of [LSZ15] aimed to achieve a similar (but weaker) fairness goal. Their idea of allowing "dropped", or "forked", blocks to be included as records inside the blockchain is clearly related to our idea of allowing "dropped" fruit to be incorporated into the chain (their protocol, however, does not have any "expiration" time for old blocks, and this feature is cruicial for achieving our goal; additionally, as mentionned, [LSZ15] only consider specific attacks and in particular does not formally prove that their protocol satisfies the desired fairness and security properties w.r.t. all attacks.)

[9]We thank Hugo Krawczyk for this phrase!

We additionally mention that the work of [GKL15] uses the idea of having two separate mining processes (and, in particular, this work introduces the "2-for-1" trick which we use); however, the end goal in [GKL15] is quite different—they employ this idea to create a byzantine agreement protocol in the permissionless setting which handles up to a fraction $\frac{1}{2}$ malicious computing power. A byzantine agreement protocol most importantly differs from a blockchain protocol in that agreement is only required for a *single* record (as opposed to a continous sequence of record). As such, a significantly simpler solution (and analysis) can be perfomed—in particular, in their approach (similarly to [LSZ15]), the equivalent of a "fruit" never needs to expire, since we only need agreement on one record. (Additionally, [GKL15] only analyze their protocol in a synchronous model of execution.)

1.3 Applications

Hybrid Consensus and Committee Election in the Permissionless Setting In both Nakamoto's blockchain protocol and ours, the time needed to confirm transactions grows with the *worst-case upper-bound* on the network delay. In contrast, in a *responsive* protocol, we require the confirmation time to only be a function of the *actual* network delay, which may be a lot smaller than the worst-case one. In a companion paper [PS16], we show how to combine any blockchain protocol with appropriate consensus algorithms to improve the latency of the blockchain protocol and achieve a responsive protocol. Roughly speaking, this "hybrid" approach (referred to as *hybrid consensus*) uses the blockchain to elect a committee—more specifically, the miners of blocks in a sufficiently long segment of the chain are elected as the committee—and then this committee executes the (standard) consensus protocol. The chain quality of the blockchain determines the fraction of honest players in the committee: if we employ Nakamoto's blockchain, we would need to require that $\frac{3}{4}$ of the computing power is controlled by honest player to ensure a chain quality of $\frac{2}{3}$ and thus a fraction $\frac{2}{3}$ honest committee members (which is required by the consensus protocol). In contrast, by relying on our new FruitChain protocol, it suffices to assume that $\frac{2}{3}$ of the computing power is controlled by honest players.

We highlight that, as shown in [PS16], achieving a reponsive protocol also *requires* assuming that $\frac{2}{3}$ of the computing is held by honest parties, and as such relying on our FruitChain protocol enables achieving an *optimal resilience* for low-latency blockchains.

More generally, using the Fruitchains protocol and such a committee election procedure, enables going from a permissionless setting with a $\rho < \frac{1}{2}$ fraction adversarial *computing power* to a permissioned setting with (close to) ρ fraction adversarial *parties* (as in standard model considered in Cryptography). Such a transformation may be useful also in other contexts.

Preventing Mining Pools An issue with the Bitcoin protocol (which relies on Nakamoto's blockchain protocol) is that the mining hardness is set so that the world (combined) finds a new block every 10 minutes—as shown in [PSS16], the mining hardness needs to be set in such a way to ensure consistency. This not only leads to a long latency (which can be remediated by the Hybrid Consensus approach discussed above), but also leads to the issue that it may take a very long time for an individual miner to be successful in mining a block and consequently reap a reward for its work. In other words, the payments received by miners has a very *high variance*. This has lead to the creation of mining pools, where miners come together and pool their work and then share the reward once someone in the pool mines a block—such pooling decreases the variance. To prevent free-riding, miners submit "partial proofs of work" (that is, "near" solutions to the mining puzzles) that are significantly easier to find, and rewards are distributed (according to some distribution rule) among the contributors of the partial proofs-of-work.

An undesirable effect of such pools is that the pool operator effectively controls a large number of participants and potentially could get them to deviate; in a sense, the decentralized nature of the system gets lost.

We note that since the FruitChain protocol is parametrized by *two* mining hardnesses—the block hardness p, and the fruit hardness p_f—which are independent of each other, we can set p appropriately to ensure consistency, but p_f can be set to be much larger—for instance, as large as the probability of find a partial proof-of-work in mining pools—and consequently, we would reduce the variance of the rewards received by miners in exactly the same way as in mining pool, but now in a *fully decentralized* way.

2 Preliminaries and Definitions

2.1 Blockchain Protocols and Executions

In this section, we recall the abstract model for blockchain protocols from [PSS16] and a provide a description of Nakamoto's original blockchain protocol which we will heavily make use of. This section is almost verbatim from [PSS16].

2.1.1 Blockchain Protocols

A blockchain protocol is a pair of algorithms $(\Pi, \mathsf{extract})$ where Π is a stateful algorithm that receives a security parameter κ as inputs and maintains a local state *chain*. The algorithm $\mathsf{extract}(\kappa, \mathit{chain})$ outputs an *ordered* sequence of "records", or "batches", $\vec{\mathsf{m}}$ (e.g., in the bitcoin protocol, each such record is an ordered sequence of transactions). We call $\mathsf{extract}(\kappa, \mathit{chain})$ the "record chain" of a player with security parameter κ and local variable *chain*; to simplify notation, whenever κ is clear from context we often write $\mathsf{extract}(\mathit{chain})$ to denote $\mathsf{extract}(\kappa, \mathit{chain})$.

(In the treatment of [PSS16], algorithm Π is also parameterized by a "validity" that encapsulates the semantic properties (e.g., "no double spending") that a blockchain application aims to achieve; we disregard those issues here as appropriate "pruning" can always be done by the higher-level application.)

A Blockchain Execution We consider the execution of a blockchain protocol $(\Pi, \mathsf{extract})$ that is directed by an environment $Z(1^\kappa)$ (where κ is a security parameter), which activates a number of parties $1, 2, \ldots, n$ as either "honest" or corrupted parties. Honest parties execute Π on input 1^κ with an empy local state *chain*; corrupt parties are controlled by an attacker A which reads all their inputs/message and sets their outputs/messages to be sent.

- The execution proceeds in *rounds* that model time steps. In round r, each honest player i receives a message (a "record") m from Z (that it attempts to "add" to its chain) and potentially receives incoming network messages (delivered by A). It may then perform any computation, *broadcast* a message to all other players (which will be delivered by the adversary; see below) and update its local state chain_i.
- A is responsible for delivering all messages sent by parties (honest or corrupted) to *all* other parties. A cannot modify the content of messages broadcast by honest players, *but it may delay or reorder the delivery of a message* as long as it eventually delivers all messages. (Later, we shall consider restrictions on the delivery time.) The identity of the sender is not known to the recipient.[10]

[10] We could also consider a seemingly weaker model where messages sent by corrupted parties need not be delivered to all honest players. We can easily convert the weaker model to the stronger model by having honest parties "gossip" all messages they receive.

- At any point, Z can communicate with adversary A or access $\mathsf{extract}(chain_i)$ where $chain_i$ is the local state of player i.
- At any point, Z can *corrupt* an honest party j which means that A gets access to its local state and subsequently, A controls party j. (In particular, this means we consider a model with "erasures"; random coin tosses that are no longer stored in the local state of j are not visible to A.)[11]
- At any point, Z can *uncorrupt* a corrupted player j, which means that A no longer controls j and instead player j starts executing $\Pi(1^\kappa)$ with a fresh state $chain_j$. (This is also how we model Z spawning a "new" honest player.) A gets informed of all such uncorrupt messages and is required to deliver all messages previously sent by (currently alive) honest players.[12]

Let $\mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa)$ be a random variable denoting the joint view of all parties (i.e., all their inputs, random coins and messages received, including those from the random oracle) in the above execution; note that this joint view fully determines the execution.

Admissible Environments We will be considering executions with restricted adversaries and environments; these restrictions will be specified by a predicate $\Gamma(\cdot,\cdot,\cdot,\cdot)$.

Definition 2.1 (Admissible Environments). *We say that the tuple* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$ *is* Γ-admissible w.r.t. $(\Pi, \mathsf{extract})$ *if A and Z are non-uniform probabilistic polynomial-time algorithms,* $\Gamma(n(\cdot), \rho, \Delta) = 1$ *and for every $\kappa \in N$, every view* view *in the support of* $\mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa)$, *the following holds:*

1. *Z activates $n = n(\kappa)$ parties in* view*;*
2. *A delays messages by at most $\Delta = \Delta(\kappa)$ rounds (and in the case of newly spawned players, instantly delivers messages that were sent more than Δ rounds ago);*
3. *at any round r in* view*, A controls at most $\rho \cdot n(\kappa)$ parties; and*

Whenever the protocol $(\Pi, \mathsf{extract})$ *is clear from context, we simply call* (n, ρ, Δ, A, Z) Γ-admissible.

2.1.2 Blockchain protocols in the ROM

To model Nakamoto's blockchain protocol, we need to extend the model with a random oracle. In an execution with security parameter κ, we assume all parties have access to a random function $H : \{0,1\}^* \to \{0,1\}^\kappa$ which they can access through two oracles: $\mathsf{H}(x)$ simply outputs $H(x)$ and $\mathsf{H.ver}(x, y)$ output 1 iff $H(x) = y$ and 0 otherwise. In any round r, the players (as well as A) may make *any* number of queries to $\mathsf{H.ver}$. On the other hand, in each round r, honest players can make only a *single* query to H, and an adversary A controlling q parties, can make q *sequential* queries to H. (This modeling is meant to capture the assumption that we only "charge" for the effort of finding a solution to a "proof of work" [DN92], but checking the validity of a solution is cheap. We discuss this further after introducing Nakamoto's protocol.) We emphasize that the environment Z does not get direct access to the random oracle (but can instruct A to make queries).

[11]Our proof actually extends also to the model "without erasures".

[12]This models the fact that a player is not considered "honest" before it has joined the network and gotten "initialized". In the real-life execution of bitcoin, new players joining send out a message to the network, request to be initialized and download the longest chain known to the network. We only consider them honest once this process is over.

2.1.3 Nakamoto's Protocol

We turn to describing Nakamoto's protocol [Nak08], which we refer to as $(\Pi^p_{Nak}, \mathsf{extract}^p_{Nak})$. The local state *chain* maintained by Π^p_{Nak} is a sequence of *(mined) blocks* $\vec{\mathsf{b}}$, where each mined block is a tuple $(h_{-1}, \eta, \mathsf{m}, h)$ that consists of a hash h_{-1} (a pointer to the previous record), a nonce η, a record m, and a hash h (a pointer to the current record[13]) and is initialized to a special "genesis" block: $(0,0,\perp), \mathsf{H}(0,0,\perp)$. Let $\mathsf{extract}(\mathit{chain})$ be the sequence of records $\vec{\mathsf{m}}$ contained in the sequence of blocks *chain*. The protocol is parameterized by a hardness function $p(\cdot)$ which defines a constant $D_p = p(\kappa) \cdot 2^\kappa$ such that for all (h, b), $\Pr_n[\mathsf{H}(h, \eta, b) < D_p] = p(\kappa)$. Whenever p is clear for context, we simply denote the protocol $(\Pi_{Nak}, \mathsf{extract}_{Nak})$ (without the p superscript); additionally, whenever κ is clear from context, we let $p = p(\kappa)$.

We say a block $\mathsf{b} = (h_{-1}, \eta, \mathsf{m}, h)$ is *valid with respect to (a predecessor block)* $\mathsf{b}_{-1} = (h'_{-1}, n', \mathsf{m}', h')$ if two conditions hold: $h_{-1} = h'$, $h = \mathsf{H}(h_{-1}, \eta, \mathsf{m})$, and $h < D_p$. A sequence of blocks $\mathit{chain} = (\mathsf{b}_0, \dots, \mathsf{b}_\ell)$ is *valid* if a) $\mathsf{b}_0 = (0, 0, \perp, \mathsf{H}(0, 0, \perp))$ is the genesis block, and b) for all $i \in [\ell]$, b_i is valid with respect to b_{i-1}.

Each round of Π_{Nak} proceeds as follows:

- Read all incoming messages (delivered by A). If any incoming message chain' is a valid sequence of blocks that is longer than its local state *chain*, replace *chain* by chain'. (Note that checking the validity of chain' can be done using only $\mathsf{H.ver}$ queries)
- Read local message m (from Z). Pick a random nonce $n \in \{0,1\}^\kappa$ and issue query $h = \mathsf{H}(h_{-1}, \eta, \mathsf{m})$ where h_{-1} is the 4'th element in the last block in *chain*. If $h < D_p$, then Π adds the *newly mined* block (h_{-1}, η, b, h) to *chain* and broadcasts the updated *chain*.

A Remark on our use of the Random Oracle Recall that in our model, we restrict players to a single evaluation query H per round, but allow them any number of verification queries $\mathsf{H.ver}$ in the same round. We do this to model the fact that checking the validity of mined blocks is "cheap" whereas the mining process is expensive. (To enable this, we have included a pointer h to the current record in every mined block in the description of Nakamoto; thus a player need not spend an H query to compute the pointer to the previous record.)

In practice, the cost of evaluating a hash function (which is used to instantiate the random oracle) is the same as verifying its outputs, but our modeling attempts to capture the phenomena that a miner typically use various heuristics (such as black lists of IP addresses that have sent invalid blocks) and different hardware to check the validity of a mined block versus to mine a new block.

2.2 Security of Blockchain Protocols

In this section, we recall the security properties of blockchains from [PSS16], which in turn are based on earlier definitions from [GKL15, KP15] For our purposes, we slightly generalize the properties from [PSS16] (see below for a discussion of this generalization), but point out that our generalized definitions suffice for all known applications of them; see [PSS16] for more discussion (and historical remarks) on these definitions.

Notation For some A, Z, consider some view in the support of $\mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa)$. We use the notation $|\mathsf{view}|$ to denote the number of rounds in the execution, view^r to denote the prefix

[13]In reality (as well as in the description in the introduction), h is not included in the block (as it can be easily determined from the remaining elements); we include it to ensure that we can verify validity of a block using only $\mathsf{H.ver}$.

of view up until round r, $chain_i(\mathsf{view})$ denotes the local state of player i in view, $\mathsf{extract}_i(\mathsf{view}) = \mathsf{extract}(chain_i(\mathsf{view}))$ and $\mathsf{extract}_i^r(\mathsf{view}) = \mathsf{extract}_i(\mathsf{view}^r)$.

Negligible Functions A function $\epsilon(\cdot)$ is said to be *negligible* if for every polynomial $p(\cdot)$, there exists some κ_0 such that $\epsilon(\kappa) \leq \frac{1}{p(\kappa)}$ for all $\kappa \geq \kappa_0$.

2.3 Chain Growth

The first desiderata is that the chain grows proportionally with the number of rounds of the protocol. Let,

$$\mathsf{min\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) = \min_{i,j} |\mathsf{extract}_j^{r+t}(\mathsf{view})| - |\mathsf{extract}_i^r(\mathsf{view})|$$

$$\mathsf{max\text{-}chain\text{-}increase}_{r,t}(\mathsf{view}) = \max_{i,j} |\mathsf{extract}_j^{r+t}(\mathsf{view})| - |\mathsf{extract}_i^r(\mathsf{view})|$$

where we quantify over players i, j such that i is honest at view^r and j is honest at view^{r+t}.

Let $\mathsf{growth}^{t_0,t_1}(\mathsf{view}, \Delta, T) = 1$ iff the following two properties hold:

- **(consistent length)** for all rounds $r \leq |\mathsf{view}| - \Delta$, $r + \Delta \geq r' \leq |\mathsf{view}|$, for every two players i, j such that in view i is honest at r and j is honest at r', we have that $|\mathsf{extract}_j^{r'}(\mathsf{view})| \geq |\mathsf{extract}_i^r(\mathsf{view})|$
- **(chain growth lower bound)** for every round $r \leq |\mathsf{view}| - t$, we have

$$\mathsf{min\text{-}chain\text{-}increase}_{r,t_0}(\mathsf{view}) \geq T.$$

- **(chain growth upper bound)** for every round $r \leq |\mathsf{view}| - t$, we have

$$\mathsf{max\text{-}chain\text{-}increase}_{r,t_1}(\mathsf{view}) \leq T.$$

In other words, $\mathsf{growth}^{t_0,t_1}$ is a predicate which tests that a) honest parties have chains of roughly the same length, and b) during any t_0 rounds in the execution, all honest parties' chains increase by at least T, and c) during any t_1 rounds in the execution, honest parties' chains increase by at most T.

Definition 2.2. *A blockchain protocol* $(\Pi, \mathsf{extract})$ *has* chain growth rate $T_0(\cdot), g_0(\cdot,\cdot,\cdot,\cdot), g_1(\cdot,\cdot,\cdot,\cdot)$ in Γ-environments *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists some negligible function* ϵ *such that for every* $\kappa \in \mathbb{N}$*,* $T \geq T_0(\kappa)$*,* $t_0 \geq \frac{T}{g_0(\kappa, n(\kappa), \rho, \Delta(\kappa))}$ *and* $t_1 = \frac{T}{g_1(\kappa, n(\kappa), \rho, \Delta(\kappa))}$ *the following holds:*

$$\Pr\left[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa) : \mathsf{growth}^{t_0,t_1}(\mathsf{view}, \Delta(\kappa), \kappa) = 1\right] \geq 1 - \epsilon(\kappa)$$

2.4 Chain Quality

Our second desideratum is that the number of records contributed by the adversary is proportional to its relative power.

We say that a *record* m *is non-adversarial (or honest) w.r.t.* view *and prefix* $\vec{\mathsf{m}}$ if there exists a player j and some round r' such that in $\mathsf{view}^{r'}$, j is honest, the environment provided m as input to j, and $\vec{\mathsf{m}}$ is a prefix of $\mathsf{extract}_i(\mathsf{view}^{r'})$. (That is, there exists some honest player that received m as an input when their chain contained $\vec{\mathsf{m}}$).

Let $\mathsf{quality}^T(\mathsf{view}, \mu) = 1$ iff for every round r and every player i such that i is honest in view^r, among any consecutive sequence of T records M in $\mathsf{extract}_i^r(\mathsf{view})$, the fraction of records m that are honest w.r.t. view^r and $\vec{\mathsf{m}}$, where $\vec{\mathsf{m}}$ is the prefix of $\mathsf{extract}_i^r(\mathsf{view})$ preceeding M, is at least μ.

Definition 2.3. *A blockchain protocol* $(\Pi, \mathit{extract})$ *has* chain quality $T_0(\cdot), \mu(\cdot,\cdot,\cdot,\cdot)$ in Γ environments, *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists some negligible function* ϵ *such that for every* $\kappa \in \mathbb{N}$ *and every* $T \geq T_0(\kappa)$ *the following holds:*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi,\mathit{extract})}(A, Z, \kappa) : \mathit{quality}^T(\mathit{view}, \mu(\kappa, n(\kappa), \rho, \Delta(\kappa))) = 1\right] \geq 1 - \epsilon(\kappa)$$

2.5 Consistency

Let $\mathsf{consistent}^T(\mathsf{view}) = 1$ iff for all rounds $r \leq r'$, and all players i, j (potentially the same) such that i is honest at view^r and j is honest at $\mathsf{view}^{r'}$, we have that the prefixes of $\mathsf{extract}_i^r(\mathsf{view})$ and $\mathsf{extract}_j^{r'}(\mathsf{view})$ consisting of the first $\ell = |\mathsf{extract}_i^r(\mathsf{view})| - T$ records are identical.[14]

Definition 2.4. *A blockchain protocol* $(\Pi, \mathit{extract})$ *satisfies* $T_0(\cdot)$-consistency in Γ environments, *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists some negligible function* ϵ *such that for every* $\kappa \in \mathbb{N}$ *and every* $T \geq T_0(\kappa)$ *the following holds:*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi,\mathit{extract})}(A, Z, \kappa) : \mathit{consistent}^T(\mathit{view}) = 1\right] \geq 1 - \epsilon(\kappa)$$

Note that a direct consequence of consistency is that the chain *length* of any two honest players can differ by at most T (except with negligible probability).

2.6 Preliminaries from [PSS16]: Security of Nakamoto's Blockchain

The results from [PSS16] (and as we shall shortly see, also ours) are parameterized by the following quantities (which are defined for some fixed mining hardness function $p(\cdot)$; recall that Nakamoto's protocol is parametrized a *single* hardness parameter p):

- let $\alpha(\kappa, n, \rho, \Delta) = 1 - (1 - p(\kappa))^{(1-\rho)n}$. That is, α is the probability that *some* honest player succeeds in mining a block in a round;
- let $\beta(\kappa, n, \rho, \Delta) = \rho n p(\kappa)$. That is β is the expected number blocks that an attacker can mine in a round.
- let $\gamma(\kappa, n, \rho, \Delta) = \frac{\alpha}{1+\Delta\alpha}$. γ is a "discounted" version of α which takes into account the fact that messages sent by honest parties can be delayed by Δ rounds and this may lead to honest players "redoing work"; γ corresponds to their "effective" mining power.

Whenever κ, n, ρ, Δ are clear from the context, we simply write p, α, β, γ. In essence, the quantities capture the per round expected "chain length increase" by the honest parties and the adversary; the reason that α, β are defined differently is that we assume that the adversary can sequentialize its queries in a round, whereas honest players make a single parallel query (they each act independently), and thus even if they manage to mine several blocks, the longest chain held by honest players can increase by at most 1. Note, however, that when p is small (in comparison to $1/n$), which is case for the Bitcoin protocol, α is well approximated by $(1-\rho)np$ and thus $\frac{\alpha}{\beta} \approx \frac{1-\rho}{\rho}$, so this difference is minor; additionally, when p is small, $\gamma \approx \alpha$ and thus $\frac{\gamma}{\beta} \approx \frac{1-\rho}{\rho}$.

Let $\Gamma_\lambda^p(n(\cdot), \rho, \Delta(\cdot)) = 1$ iff $n(\cdot), \Delta(\cdot)$ are polynomially-bounded functions $\mathbb{N} \rightarrow \mathbb{N}^+$, $0 \leq \rho \leq 1$ and for all $\kappa, n = n(\kappa), \Delta = \Delta(k)$,

$$\alpha(1 - 2(\Delta + 1)\alpha) \geq \lambda\beta$$

As in show in [PSS16], this condition implies the following condition.

[14]Pedantically, the "first ℓ records of $\mathsf{extract}_j^{r'}(\mathsf{view})$ is not defined if $\mathsf{extract}_j^{r'}(\mathsf{view}) < \ell$; to formalize it, we may represent the chains as infinite sequences of records, where all records after the end of the chain is a special "nil" symbol. In particular, this ensures that $\mathsf{consistent}^T(\mathsf{view}) = 0$ if $\mathsf{extract}_j^{r'}(\mathsf{view}) < \ell$.

Fact 2.5. *If $\Gamma_\lambda^p(n(\cdot), \rho, \Delta(\cdot)) = 1$ then $np\Delta < 1$.*

We directly get the following corollary that will be useful to us.

Fact 2.6. *If $\Gamma_\lambda^p(n(\cdot), \rho, \Delta(\cdot)) = 1$ then $\gamma \geq \frac{np}{8}$ and*

Proof. Recall that $\gamma = \frac{\alpha}{1+\Delta\alpha}$. Since $\alpha \leq np$, by Fact 2.5, we directly get that

$$\gamma \geq \frac{\alpha}{2}\beta$$

Recall that, $\alpha = 1 - (1-p)^{(1-\rho)n}$. Since by Fact 2.5, $n < 1/p$, by the binomial expansion we have that

$$(1-p)^{(1-\rho)n} < 1 - \frac{(1-\rho)np}{2}$$

Thus, $\gamma > \frac{(1-\rho)np}{4} \geq \frac{np}{8}$ since under the Γ restriction $\rho < \frac{1}{2}$. $\square$

The following theorem was proven in [PSS16].

Theorem 2.7 (Security of Nakamoto [PSS16])**.** *For any $\delta > 0$, any $\lambda > 1$, any $p(\cdot)$, any superlogarithmic function $T_0(\cdot)$, $(\Pi_{Nak}^p, \mathsf{extract}_{nak}^p)$ satisfies:*

- *T_0-consistency;*
- *chain growth rate $(T_0, g_0^{p,\delta}, g_1^{p,\delta})$ where*

$$g_0^{p,\delta}(\kappa, n, \rho, \Delta) = (1-\delta)\gamma$$
$$g_1^{p,\delta}(\kappa, n, \rho, \Delta) = (1+\delta)np$$

- *chain quality $T_0, \mu_\delta^p(\kappa, n, \rho, \Delta)$ where*

$$\mu_\delta^p(\kappa, n, \rho, \Delta) = 1 - (1+\delta)\frac{\beta}{\gamma};$$

in Γ_λ^p environments.

Remark 2.8 (Blockchain quality and consistency)**.** *The consistency property proven in [PSS16] is actually a bit stronger than stated. Not only it shows that players agree on the* records *contained in their blockchains, but also that the actual* blockchains *agree except for potentially the last κ blocks. We refer to this property as* blockchain consistency*, and will rely on it in the sequel.*

Additionally, the chain quality property is also stronger in that not only the records of honest block are contributed by honest players, but also the actual block are mined by honest players. We refer to this property as blockchain quality*, and will rely on it in the sequel.*

2.7 Liveness

The liveness property from [PSS16] (which generalized the one from [GKL15]), stipulates that from any given round r, if a sufficiently long period of time t elapses—we refer to this time as the *wait-time* of the blockchain—*every* honest player will have a message m sufficiently "deep" in their chain (technically, κ blocks from the end of the chain), where m was provided as an input to some honest player between rounds r and $r+t$[15] More precisely, let $\mathsf{live}(\mathsf{view}, t) = 1$ iff for any t consecutive rounds $r, \ldots, r+t$ in view there exists some round r' s.t. $r \leq r' \leq r+t$ and player i such that in view, 1) i is honest at r', 2) i received a message m as input at round r', and 3) for every player j that is honest at $r+t$ in view, $\mathsf{m} \in \mathsf{extract}_j^{r+t}(\mathsf{view})[: -\kappa]$.

[15]The weaker liveness property from [GKL15] only requires this is *all* honest players have m as their input; this weaker property is not enough for our purposes.

Definition 2.9. *We say that blockchain* $(\Pi, \mathsf{extract})$ *satisfies* liveness with wait-time $w(\cdot,\cdot,\cdot,\cdot)$ in Γ environments *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, there exists a negligible function* ϵ *in the security parameter* $\kappa \in \mathbb{N}$*, such that*

$$\Pr\left[\mathsf{view} \leftarrow \mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa) : \mathsf{live}(\mathsf{view}, w(\kappa, n(\kappa), \rho, \Delta(\kappa)) = 1\right] \geq 1 - \epsilon(\kappa)$$

The following theorem was shown in [PSS16].

Theorem 2.10 ([PSS16]). *Let* $(\Pi, \mathsf{extract})$ *be a blockchain protocol satisfying chain growth* T_0, g_0, g_1*, chain quality* T_0, μ *and* T_0*-chain consistency in* Γ*-environments, where* μ *and* g_0 *are strictly positive, and* T_0 *is sublinear.*[16] *Then,* $(\Pi, \mathsf{extract})$ *satisfies liveness with wait-time*

$$w(\kappa, n, \rho, \Delta) = (1+\delta)\frac{\kappa}{g_0(\kappa, n, \rho, \Delta)}$$

in Γ*-environments.*

As a direct corollary of 2.10 and 2.7, we get:

Corollary 2.11 ([PSS16]). *For any* $\lambda > 1$*,* $\delta > 0$*, and any* $p(\cdot)$*,* $(\Pi^p_{\mathsf{Nak}}, \mathsf{extract}^p_{\mathsf{Nak}})$ *satisfies liveness with wait-time*

$$w(n, \kappa, \rho, \Delta) = (1+\delta)\,\frac{\kappa}{\gamma}$$

in Γ^p_λ *environments.*

3 Defining Fairness

We turn to defining our notion of fairness. Roughly speaking, a blockchain protocol is δ-*approximately fair* w.r.t. ρ attackers if, with overwhelming probability, any $\phi \leq 1 - \rho$ fraction coalition of *honest* users is guaranteed to get at least a

$$(1-\delta)\phi$$

fraction of the blocks in every sufficiently long window, even in the presense of an adversary controlling a ρ fraction of the computation power. Note that this condition trivially implies $(1-\delta)(1-\rho)$ chain quality (by considering $\phi = 1 - \rho$, that is, the full set of honest players). Consequently, to formally define this notion, we first generalize the definition of $\mathsf{quality}$ (used in the definition of chain quality, see Definition 2.3) to consider "quality" w.r.t to any subset S of the honest players; for generality, we allow the subset to change over time.

- Let a *player subset selection*, S, be a function that given a view view outputs a subset of the players that are honest at view.
- We say that S is a ϕ*-fraction player subset selection* if $S(\mathsf{view})$ always outputs a set of size ϕn (rounded upwards) where n is the number of players in view.
- Given an player subset selection S, we say that a *record* m *is* S*-compatible w.r.t.* view *and prefix* $\vec{\mathsf{m}}$ if there exists a player j and some round r' such that j is in $S(\mathsf{view}^{r'})$, the environment provided m as an input to j at $\mathsf{view}^{r'}$, and $\vec{m}$ is a prefix of $\mathsf{extract}^{r'}_i(\mathsf{view})$.

[16] [PSS16] only explictly considered the case when T_0 is some slightly super-logarthmic function, but their proof actually only assumes that T_0 is sublinear. We also remark that any blockchain protocol which satisfies the security properties w.r.t. to a polynomial T_0 which potentially is super-linear can always be modified to satisfy security w.r.t. a sublinear T_0 by redefining the security parameter.

- Let $\mathsf{quality}^{T,S}(\mathsf{view}, \mu) = 1$ iff for every round r and every player i such that i is honest in view^r, we have that among any consecutive sequence of T records M in $\mathsf{extract}_i^r(\mathsf{view})$, the fraction of records m that are S-compatible w.r.t. view^r and $\vec{m}$, where $\vec{\mathsf{m}}$ is the prefix of $\mathsf{extract}_i^r(\mathsf{view})$ preceeding M, is at least μ.

We now define fairness analogously to chain quality.

Definition 3.1. *A blockchain protocol* $(\Pi, \mathit{extract})$ *has (approximate)* fairness $T_0(\cdot), \delta$ in Γ environments, *if for all* Γ*-admissible* $(n(\cdot), \rho, \Delta(\cdot), A, Z)$*, every* $\phi \leq 1 - \rho$*, every* ϕ*-fraction subset selection* S*, there exists some negligible function* ϵ *such that for every* $\kappa \in \mathbb{N}$ *and every* $T \geq T_0(\kappa)$ *the following holds:*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi,\mathit{extract})}(A, Z, \kappa) : \mathit{quality}^{T,S}(\mathit{view}, (1-\delta)\phi)) = 1\right] \geq 1 - \epsilon(\kappa)$$

As a sanity check, note that the definition of $\mathsf{quality}^{T,S}(\mathsf{view}, \mu)$ collapses down to $\mathsf{quality}^{T}(\mathsf{view}, \mu)$ if S is the full set of the honest players. As a consequence, (T_0, δ)-fairness trivially implies $(T_0, (1-\delta)(1-\rho))$-chain quality (by considering $\phi = 1 - \rho$). Additionally, when $\rho \leq \frac{1}{2}$ (which is the case we consider in this paper,

$$(1-\delta)(1-\rho) = 1 - \delta - \rho + \delta\rho = 1 - [\delta + (1-\delta)\rho] \geq 1 - [2\delta\rho + (1-\delta)\rho] = 1 - (1+\delta)\rho$$

Thus, no ρ-size coalition can get more than a factor $(1+\delta)$ more than its "fair" share of blocks.

Fact 3.2. *If a blockchain protocol* $(\Pi, \mathit{extract})$ *satisfies* $(T_0(\cdot), \delta)$*-fairness in* Γ *environments, then it satisfies* (T_0, μ)*-chain quality, where* $\mu(\kappa, n, \rho, \Delta) = (1-\delta)(1-\rho) \geq 1 - (1+\delta)\rho$ *when* $\rho \leq \frac{1}{2}$.

4 The FruitChain Protocol

We now turn to formally defining our FruitChain protocol. Rougly speaking, the FruitChain protocol will be running an instance of Π^p_{Naka} but instead of directly putting the records m inside the blockchain, the records are put inside "fruit" f; these fruit themselves requires solving some proof of work—with a *differerent hardness parameter* p_f; additionally, we require the fruit to "hang" from a block which isn't too far from from the block which records the fruit—more specifically, the fruit needs to "point" to an earlier block in the chain which is not too far from the block containing it (and thus, the fruit could not have been mined "too" long ago); the *recency parameter* R will be used to specify how far back a fruit is allowed to hang.

Towards formalizing the protocol, we start by introducing some notation:

- We assume that the random oracle H outputs strings of length at least 2κ. Let d be a collision-resistant hash-function (technically, it is a family of functions, and the instance from the family is selected as a public-parameter; in the sequel we ignore this selection and simply treat it as a single function (for instance, selected using randomness $H(0)$.)

- Our protocol is parametrized by two "hardness" parameters $p = p(\kappa), p_f = p_f(\kappa)$, and a recency parameter R. (p is the mining hardness parameter for Π^p_{Naka} and p_f is the "fruit mining" hardness parameter, as mentionned above, the recency parameter will specify how far back a fruit is allowed to "hang"); the quantity $q = q(\kappa) = \frac{p_f(\kappa)}{p(\kappa)}$ will be useful in our analysis.

- We say that a *fruit,* $f = (h_{-1}; h'; \eta, \mathsf{digest}; \mathsf{m}; h)$*, is valid* iff $H(h_{-1}; h'; \eta; \mathsf{digest}; \mathsf{m}) = h$ and $[h]_{-\kappa:} < D_{p_f}$ where $[h]_{-\kappa:}$ denotes the last κ bits of h; we call h' the $\mathsf{pointer}$ of f. *F is a valid fruit-set* if either $F = \emptyset$ or F is a set of valid fruits.

- We say that a *block*, $\mathsf{b} = ((h_{-1}; h'; \eta; \mathsf{digest}; m; h), F)$, *is valid* iff $\mathsf{digest} = \mathsf{d}(F)$, F is a valid fruit-set, $H(h_{-1}; h'; \eta, \mathsf{d}(F); \mathsf{m}) = h$ and $[h]_{:\kappa} < D_{p_1}$ where $[h]_{:\kappa}$ denotes the first κ bits of h; we call h the reference of b.

- We say that a *sequence of blocks, chain* $= (\mathsf{b}_0, \ldots, \mathsf{b}_\ell)$, *is valid* where $\mathsf{b}_i = ((h^i_{-1}; h'^i; \eta^i, \mathsf{digest}^i; \mathsf{m}^i; h^i), F^i)$ iff
  - $\mathsf{b}_0 = genesis$ where $genesis := ((0; 0; 0; 0; \perp; \mathsf{H}(0; 0; 0; 0, \perp)), \emptyset)$ is the "genesis" block;
  - for all $i \in [\ell]$, $h^i_{-1} = h^{i-1}$,
  - for all $i \in [\ell]$, all $f \in F^i$, there exists some $j \geq i - R\kappa$ such that the pointer of f is h^j.

- finally, we say that the *fruit* f *is* recent *w.r.t. chain* if the pointer of f is the reference of a block in $chain[-R\kappa :]$ (i.e., one of the last $R\kappa$ blocks in $chain$).

The FruitChain protocol is described in Figure 1.

We are now ready to state our main theorem.

Theorem 4.1. *For any $0 < \delta < 1$, any $\lambda > 1$, and any $p(\cdot), p_f(\cdot)$, let $R = 17$, $\kappa_f(\kappa) = 2q(\kappa)R\kappa$, and $T_0(\kappa) = 5\frac{\kappa_f}{\delta}$. Then $(\Pi^{p,p_f,R}_{fruit}, \mathsf{extract}^{p,p_f,R}_{fruit})$ satisfies:*

- *κ_f-consistency;*

- *chain growth rate $(T_0, g^{p,\delta}_0, g^{p,\delta}_1)$ where*

$$g^{p,\delta}_0(\kappa, n, \rho, \Delta) = (1 - \delta)(1 - \rho)np_f,$$

$$g^{p,\delta}_1(\kappa, n, \rho, \Delta) = (1 + \delta)np_f$$

- *fairness (T_0, δ).*

in Γ^p_λ environments.

5 Proof of the Main Theorem

We start by introducing some additional notation and useful lemmas, and then turn to proving each of the three security properties.

5.1 Additional Notation

Let us introduce some additional notation that will be useful in the analysis of the protocol:

- We say that a fruit $f = (h_{-1}; h'; \eta, \mathsf{digest}; \mathsf{m}; h)$ was *mined* at round r if r is the first time H outputs h.

- We say that a block, $\mathsf{b} = ((h_{-1}; h'; \eta; \mathsf{digest}; \mathsf{m}; h), F)$ was *mined* at round r if r is the first time H outputs h.

- We say that a block/fruit was mined by an honest player if there it was an honest players that first mined it.

To simplify notation, in addition to the parameters α, β, γ previously defined, we also define analogs of α and β with respect to the "fruit mining" process: let,

- $\alpha_f = (1-\rho)np_f$ (that is, the expected number of fruit mined by honest players in one round);

- $\beta_f = \rho np_f$ (that is, the expected number of fruit mined by honest players in one round).

Π_{fruit}: FruitChain **protocol**

```
**Initialize:** $chain := genesis$, $F = \emptyset$

Upon receiving a valid $fruit$,
    • let $F := F \cup \{fruit\}$

Upon receiving a valid $chain'$, if $|chain'| > |chain|$:
    • let $chain := chain'$

Every time step, upon receiving input $\mathsf{m}$ from the environment:
    • let $F'$ be all fruits $f \in F$ that are recent w.r.t. $chain$;
    • let $h'$ be the reference of $chain[pos]$ where $pos = max(1, |chain| - \kappa)$;
    • let $h_{-1}$ be the reference of $chain[-1]$;
    • Pick random $\eta \in \{0,1\}^\kappa$ and let $h := \mathsf{H}(h_{-1}; h'; \eta; \mathsf{d}(F'); \mathsf{m})$
    • If $[h]_{-\kappa:} < D_{p_f}$ (i.e., we "mined a fuit")
        – let $fruit := (h_{-1}; h'; \eta; \mathsf{d}(F'); \mathsf{m}, h)$, $F := F \cup \{fruit\}$, and broadcast $fruit$
    • If $[h]_{:\kappa} < D_p$ (i.e., we "mined a block")
        – let $chain := chain||((h_{-1}; h'; \eta, \mathsf{d}(F'); \mathsf{m}, h), F)$, and broadcast $chain$
```

extract$_{fruit}$:

On input a valid chain *chain*, output the sequence of messages m contained in the fruit contained in blocks of *chain*, ordered by:

- the block (which contained the fruit, which contains the message)—that is, messages inside fruits inside earlier blocks, come earlier;
- in the case of ties (i.e., if some block contains multiple fruits), break ties by the pointer of the fruit (giving preference to fruit pointing to earlier blocks), and finally if have the same pointer, just lexicographically.[a]

[a]The method for how we break times among fruit in the *same* block is inconsequential for our results.

Figure 1: The FruitChain protocol. Nodes not only mine for blocks, but also fruit. Blocks confirm "recent" fruit; whereas fruit confirm transactions.

5.2 The Fruit Freshness Lemma

In this section, we present a lemma demonstrating the key property of the FruitChain protocol: any fruit mined by an honest player will be incorporated sufficiently deep in the chain (and thus never lost). We refer to this as the *Fruit Freshness Lemma*—fruits stay "fresh" (i.e., recent) sufficiently long to be incorporated.

Let $\mathsf{fruitfreshness}(\mathsf{view}, w, \kappa) = 1$ iff for every honest player i and every round $r < |view| - w$, if i mines a fruit at round r in view, then for every honest player j, there exists some index pos such that f is at position pos in the record chain (w.r.t. Nakamoto's protocol) of j at every round $r' \geq r + w$ (i.e., $f \in \mathsf{extract}^{r'}_{naka,j}(\mathsf{view})[pos]$) and additionally pos is at least κ positions from the end of the chain.

Let

$$wait(\kappa, n, \rho, \Delta) = 2\Delta + \frac{2\kappa}{\gamma}$$

Lemma 5.1. *For any $\lambda > 1$, any ρ', any $p(\cdot)$, $p_f(\cdot)$, any Γ^p_λ-admissible $(n(\cdot), \rho, \Delta(\cdot), A, Z)$, there exists a negligible function ϵ such that*

$$\Pr\left[\mathit{view} \leftarrow \mathit{EXEC}^{(\Pi^{p,p_f,R},\mathit{extract}^{p,p_f,R})}(A, Z, \kappa) : \mathit{fruitfreshness}(\mathit{view}, wait(\kappa, n(\kappa), \rho, \Delta(\kappa)), \kappa) = 1\right] \geq 1 - \epsilon(\kappa)$$

when $R = 17$.

Proof. Disregard the blockchain consistency (see Remark 2.8), liveness and chain growth failure events—they only happen with negligible probability. Let $wait = wait(\kappa, n, \rho, \Delta)$.

- By *blockchain consistency*, at any point in the execution, whenever an honest player mines a fruit f, the block pointed to by the fruit is at some *fixed* position pos on the blockchain of every honest player, now and at every time in the future. (Recall that honest players try to mine fruit that point back to a block that is κ steps back in the chain, and thus the consistency condition kicks in.) Let ℓ denote the length of the chain of the player that mines f; by definition $pos = \ell - \kappa$.

- By the description of the protocol, if the fruit f is mined at a round r', it gets seen by all honest players by round $r' + \Delta$; additionally, when this happens all honest players attempt to add f to their chain as long as it remains recent (w.r.t. all honest players).

- By *liveness*, it thus follows that f gets incorporated into the record chain of all honest players at some position pos that is at least κ records from the end of their chain by round

$$r' + \Delta + (1 + \delta)\frac{\kappa}{\gamma} \leq r' + wait - \Delta$$

  as long as f is recent by then (w.r.t. all honest players).

- By the upperbound on chain growth, at most

$$(1 + \delta)np\left(\Delta + \frac{2\kappa}{\gamma}\right)$$

  blocks are "added" in time $wait - \Delta$; more precisely, by round $r' + wait - \Delta$, no honest player has ever had a chain of length ℓ' such that

$$\ell' > \ell + (1 + \delta)np\left(\Delta + \frac{2\kappa}{\gamma}\right)$$

Thus, by round $r' + wait - \Delta$, for every such honest player's chain length ℓ' we have

$$pos = \ell - \kappa \geq \ell' - \kappa - (1+\delta)np\left(\Delta + \frac{2\kappa}{\gamma}\right)$$

By our assumption on Γ and by Fact 2.5 and Fact 2.6, we have that $\gamma \geq \frac{np}{8}$ and $np\Delta < 1$, thus

$$pos \geq \ell' - \kappa - (1+\delta) - (1+\delta)16\kappa \leq 17\kappa = \ell' - R\kappa$$

which means that f remains recent until round $r' + wait - \Delta$ w.r.t. all honest players.

- Finally, by *consistency*, all honest players agree that f is found at position *pos* in their blockchain at any point after $r' + wait - \Delta$; additionally, by the *consistent lenght property* all honest players agree that position *pos* is at least κ from the end of the chain by $r' + wait - \Delta + \Delta = r' + wait$.

□

We also observe the following fact about *wait*, which says that the expected number of fruits mined by all players during $wait + 2$ steps is upper bounded by k_f.

Fact 5.2. *For any $\lambda > 1$, any ρ', any $p(\cdot)$, $p_f(\cdot)$, any Γ^p_λ-admissible $(n(\cdot), \rho, \Delta(\cdot), A, Z)$,*

$$(wait + 2) \cdot np_f \leq k_f$$

Proof. Note that by Fact 2.5 and Fact 2.6, we have that $\gamma \geq \frac{np}{8}$ and $np\Delta < 1$, thus

$$(wait + 2) \cdot np_f = (2\Delta + 2\frac{\kappa}{\gamma} + 2) \cdot qpn \leq 2q + 2\kappa \cdot 8q + 2 \leq 2qR\kappa = \kappa_f$$

□

5.3 Some simplifying assumptions

Towards, proving our main theorem we state some simplyfing assumptions that can be made without loss of generality. These assumptions (which all follow from properties of the random oracle H) will prove helpful in our subsequent analysis.

- **WLOG1**: We may without loss of generality assume that honest players *never* query the RO on the same input—more precisely, we analyze an experiment where if some honest player wants to query it on an "old" input, it resamples nonce until the input is "new"; since nonce is selected from $\{0,1\}^\kappa$, this "resampling" experiment is identical to the real one with except with negligible probability, thus we can wlog analyze it.

- **WLOG2**: We may without loss of generality assume that any fruit that points to a block b which was first mined at time t, has been mined after t. Additionally, any fruit that points to a block that comes after b in a valid chain must have been mined after t. (If not, we can predict the outcome of the random oracle H on some input before having queries H which is a contradiction. We omit the standard details.)

- **WLOG3**: We may assume without loss of generality that all fruit mined by honest players are "new" (i.e., different from all valid fruit previously seen by honest players); this follows by WLOG1 and the fact that the probability of seeing a collision in the random oracle is negligible (by a simple union bound over the number of random oracle queries).

- **WLOG4**: We may assume without loss of generality that any valid fruit which appears in some honest players chain at round r was mined before r; this follows from the unpredictability of the random oracle (and a simple union bound over the number of random oracle queries).
- **WLOG5**: We may assume without loss of generality that there are no "blockchain collisions"—namely, there are no two *different* valid sequences of blocks which end with the same block.

We now turn to proving the three security properties.

5.4 Proof of Fruit Consistency

Disregard the chain growth and consistency, and blockchain quality (see Remark 2.8) failure events—they happen with negligible probability. Consider some view view in the support of $\mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa)$, rounds r, r' s.t. $r' \geq r$, and players i, j that are honest respectively at r, r' in view. By *consistency*, the chains of i, j at r, r' agree except for potentially the last κ blocks in the chain of i—let $C = \mathsf{b}_0, \ldots, \mathsf{b}_{|C|}$ denote those initial blocks on which they agree, and let $\mathsf{b}_{|C|+1}, \ldots$ denote the (max κ) blocks in the chain of i at r which are not in the chain of j at r'; we now bound the number of fruits that can be contained in these remaining (max κ) "inconsistent" blocks.

- By the "recency condition" of valid fruit, any valid fruit in the chain of i at r which is after C must point to a block $\mathsf{b}_{j'}$ such that $j' > |C| - R\kappa$.
- By the *blockchain quality condition*, there exists some j'' s.t. $|C| - R\kappa - \kappa \leq j'' \leq |C| - R\kappa$ and $\mathsf{b}_{j''}$ was mined by an honest player. Let r'_0 denote the round when this block was mined.
- Note that at r'_0, $\mathsf{b}_{j''}$ was mined by an honest player holding a chain of length $j'' \geq |C| - R\kappa - \kappa$; additionally, at r, i is honest, holding a chain of length at most $|C| + \kappa$ (recall that $|C|$ contains the blocks on which i and j agree, and by consistency, all but the last κ blocks in the chain of i must be in the chain of j). Thus, by the *chain growth upperbound*, at most

$$\mu = (1+\delta)\frac{2\kappa + R\kappa}{np}$$

  rounds could thus have elapsed between r'_0 and r.
- By WLOG2, any fruit which gets added after C must have been mined after r'_0. By WLOG4, any such fruit that is part of the chain of i by r was mined before r.
- We thus conclude by the Chernoff bound (see Lemma A.1) that for every sufficiently small δ', except with probability $e^{-\Omega(np_f \cdot \frac{\kappa(R+2)}{np})} = e^{-\Omega(q(R+2)\kappa)}$, there were at most

$$(1+\delta')^2 \cdot np_f \cdot \frac{\kappa(R+2)}{np} = (1+\delta')^2 q(R+2)\kappa < 2qR\kappa = \kappa_f$$

  "inconsistent" fruit in the chain of i at r.

5.5 Proof of Fruit Growth

Consistent Length The consistent length property follows directly from the consistent length property of the underlying blockchain.

The Lowerbound Disregard the fruit freshness failure event (Lemma 5.1)—it happens with negligible probability. Consider any r, t and players i, j that are honest respectively at round r and $r + t$. Consider the t rounds starting from round r.

- By the *fruit freshness* condition, every fruit that is mined by some honest party by round $r + t - wait$ gets incorporated into (and remains in) the chain of player j by $r + t$.
- By the Chernoff bound, in the $t - wait$ rounds from r to $r + t - wait$, except with probability $e^{-\Omega((t-wait)\alpha_f)}$, the honest parties mine at least

  $$(1 - \delta')(t - wait)\alpha_f$$

  fruits (where δ' is some arbitrarily small constant), which are all included in the chain of j at $r + t$. Additionally, by WLOG3 they are all "new" (i.e., not included in the chains of i at r) and different.
- Finally, by *fruit consistency* (proved in Section 5.4), we have that all but potentially κ_f of the fruits in the chain of i at r are still in the chain of j at $r + t$.
- We conclude that, except with probability $e^{-\Omega((t-wait)\alpha_f)}$, the chain of j at $r + t$ contains at least

  $$(1 - \delta')(t - wait)\alpha_f - \kappa_f$$

  more fruits than the chain of j at r. By Fact 5.2, $wait \cdot \alpha_f = wait \cdot (1-\rho)np_f \leq (1-\rho)\kappa_f \leq \kappa_f$; thus, have at least

  $$(1 - \delta)(t - wait)\alpha_f - \kappa_f \geq (1 - \delta)\alpha_f t - 2\kappa_f \qquad (1)$$

  new fruit.

We conclude by noting that this implies that implies a fruit growth lowerbound of $g_0 = \frac{1}{1+\delta}\alpha_f \geq (1 - \delta)\alpha_f$ in the desired regime: Consider any $T \geq \frac{5\kappa_f}{\delta}$ and any

$$t \geq \frac{T}{g_0} = \frac{T}{\frac{\alpha_f}{1+\delta}}$$

As shown above (see Equation 1), during this time t, except with probability, $e^{-\Omega((t-wait)\alpha_f)}$ the chain must grown by at least

$$T(1 + \delta)(1 - \delta') - 2\kappa_f = T(1 + \frac{\delta}{2})(1 - \delta') + T\frac{\delta}{2}(1 - \delta') - 2\kappa_f$$

For a sufficiently small δ' the first term is greater than T, and the second term greater than $2\kappa_f$, and thus the chain must have grown by at least T. Finally note that by Equation 1

$$e^{-\Omega((t-wait)\alpha_f)} = e^{-\Omega((t\alpha_f-\kappa_f)} = e^{-\Omega((T-\kappa_f)} = e^{-\Omega((5\kappa_f-\kappa_f)} = e^{-\Omega((5\kappa_f-\kappa_f)} = e^{-\Omega(\kappa)}$$

Thus the chain growth is guaranteed except with negligible probability.

Upperbound Disregard the chain growth, consistency and blockchain quality (see Remark 2.8) failure events—they happen with negligible probability. Consider some view view in the support of $\mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa)$, rounds $r, r' = r + t$ and players i, j that are honest respectively at r and r' in view. By *consistency*, the chains of i, j at r, r' agreee except for potentially the last κ blocks of the chain of i—let $C = \mathsf{b}_0, \ldots, \mathsf{b}_{|C|}$ denote those initial blocks on which they agree, and let $\mathsf{b}_{|C|+1}, \ldots$ denote the blocks in the chain of j at r' which are not in the chain of i at r (there may be more than κ such blocks since we are looking at the chain of j at a later time r'); We now upper bound the number of fruits in the new blocks in the chain of j which come after C, similarly to the fruit consistency proof (they main difference is that we now consider the chain of j as opposed to the chain of i). The details follow:

- By the "recency condition" of valid fruit, any valid fruit in the chain of j at r' which is after C must point to a block $\mathsf{b}_{j'}$ such that $j' > |C| - R\kappa$,
- By the *blockchain quality condition*, there exists some j'' s.t. $|C| - R\kappa - \kappa \leq j'' \leq |C| - R\kappa$ and $\mathsf{b}_{j''}$ was mined by an honest player. Let r'_0 denote the round when this block was mined.
- Note that at r'_0, $\mathsf{b}_{j''}$ was mined by an honest player holding a chain of length $j'' \geq |C| - R\kappa - \kappa$; additionally, at r, i is honest, holding a chain of length at most $|C| + \kappa$ (recall that $|C|$ contains the blocks on which i and j agree, and by consistency, all but the last κ block in the chain of i must be in the chain of j). Thus, by the *chain growth upperbound*, for any arbitrarily small δ' at most
$$\mu = (1+\delta')\frac{2\kappa + R\kappa}{np}$$
rounds could thus have elapsed between r'_0 and r.
- By WLOG2, any fruit which gets added after C must have been mined after r'_0. By WLOG4, any such fruit that is part of the chain of j by r' was mined before r'.
- We thus conclude by the Chernoff bound that except with probability $e^{-\Omega(np_f \cdot \frac{\kappa(R+2)}{np})} = e^{-\Omega(q(R+2)\kappa)}$, there were at most
$$(1+\delta')^2 \cdot np_f \cdot \left(\frac{\kappa(R+2)}{np} + t\right) = (1+\delta')^2 \left(q(R+2)\kappa + np_f t\right) \leq \kappa_f + (1+\delta')^2 np_f t \qquad (2)$$
"new" fruits in the chain of j at r'.

We conclude by noting that this implies a fruit growth upperbound of $g_1 = (1+\delta)np_f$ in the desired regime: Consider any $T \geq \frac{5\kappa_f}{\delta}$ and any
$$t = \frac{T}{g_1} = \frac{T(1+\delta)}{np_f}.$$
As shown above (see Equation 2), during this time t, except with negligible probability, the chain must have grown by at most
$$\kappa_f + (1+\delta')^2 T(1-\delta) \leq T\delta/5 + (1+\delta')^2 T/(1+\delta)$$
For any $0 < \delta < 1$ and $\delta' = 0.1\delta$, the above expression is upper bounded by T.

5.6 Proof of Fruit Fairness

Disregard the chain growth, blockchain quality (see Remark 2.8), fruit freshness, and the fruit growth failure events—they happen with negligible probability. Consider some ϕ-fraction player subset selection S, some view view in the support of $\mathsf{EXEC}^{(\Pi,\mathsf{extract})}(A, Z, \kappa)$, some round r and player i that is honest at view^r. Let $C = \mathsf{b}_0, \ldots, \mathsf{b}_{|C|}$ be the blocks in the view of i at view, let $f_0, \ldots, f_\ell$ be the fruits contained in them, and let $\mathsf{m}_0, \ldots, \mathsf{m}_\ell$ be the records contained in the fruits; let $f_j, \ldots, f_{j+T}$ be T consecutive fruits for some j, where $T \geq \frac{5\kappa_f}{\delta}$

Let r_0 be the round when the *block* in the view of i at r containing $f_{j+\kappa_f}$ was first added to *some* honest player j_0's chain[17]; let r_1 be the round when the block (again in the view of i at r) containing f_{j+T} was first added to some honest player j_1's chain, and let $t = r_1 - r_0 - 2$ be the number of rounds from $r_0 + 1$ to $r.1 - 1$. We lower bound the number of S-compatible (honest) fruits in the sequence, following similar lines (but slightly more complicated) to the proof of fruit growth *lowerbound*:

- By the *fruit freshness* condition, every fruit mined by some honest player between $(r_0 + 1)$ and $(r_1 - 1) - wait$ will be in the chain of j_1 at some position *pos* that is at least κ positions from the end of the chain, before the beggining of round r_1 and will remain so.
- By the Chernoff bound, in the $t - wait$ rounds from $r_0 + 1$ to $(r_1 - 1) - wait$, except with probability $e^{-\Omega((t-wait)\phi np_f)}$, the honesst parties in S mine at least

$$(1 - \delta')(t - wait)\phi np_f$$

  fruits (where δ' is some arbitrarily small constant), which thus are all included in the chain of j_1 by $r_1 - 1$.
- Since fruit are ordered by the block containing them, and since in round r_1 a *new* block is added which contains f_{j+T}, it follows from *blockchain consistency* that all these fruit are contained in the sequence $f_1, \ldots, f_{j+T}$ (recall that all these fruit are found in blocks that are at least κ positions from the end of the chain, so by consistency, those block cannot change and thus were not added in round r_1 and consequently must come before the block containing f_{j+T}).
- By WLOG3, these fruit are also all "new" (i.e., not included in the chains of j_0 at r_0) and different. Since in round r_0, the block containing f_j was added to the chain of j_0, and since by WLOG5, the chain of j_0 at r_0 up until (and including) the block which contains f_j is a prefix of C, all these fruit must in fact be contained in the sequence $f_{j+\kappa_f}, \ldots, f_{j+T}$.
- Finally, by fruit consistency, at r_0 all honest players' fruit chain contain $f_1, \ldots f_j$ (since recall that some player added $f_{j+\kappa_f}$ at r_0. Thus all these fruits are S-compatible w.r.t the prefix $f_1, \ldots f_{j-1}$ before the T segment we are considering.

We proceed to show that t is sufficiently large. Recall that j_0 is honest at r_0 and j_1 is honest at r_1. We know that at r_1, the fruit chain contains at least f_{j+T} fruit. Additionally, at r_0 the fruit $f_{j+\kappa_f}$ is added for the first time, so by fruit chain consistency, at most $j + 2\kappa_f$ fruit could have been in the chain of i at this point (since a fruit at position j is modified). Thus, the fruit chain must have grown by at least $T - 2\kappa_f$ from r_0 to r_1. By the *upperbound on fruit growth* (see Equation 2) we thus have that

$$T - 2\kappa_f \le \kappa_f + (1 - \delta')^2 np_f(t + 2)$$

Thus,

$$t \ge \frac{1}{(1 + \delta')^2 np_f}(T - 3\kappa_f) - 2$$

[17]Note that we cannot consider the time when it was added to i's chain as i may potentially be corrupted up until r.

We conclude that (except with negligible probability) the number of fruits in the sequence is at least:

$$\begin{aligned}
(1-\delta')\phi n p_f \left(\frac{1}{(1+\delta')^2 n p_f}(T-3\kappa_f) - 2 - \mathit{wait}\right) &= \\
(1-\delta')\phi \left(\frac{1}{(1+\delta')^2}(T-3\kappa_f) - n p_f(\mathit{wait}+2)\right) &\geq \\
(1-\delta')\phi \left(\frac{1}{(1+\delta')^2}(T-3\kappa_f) - \kappa_f\right) &\geq \\
(1-\delta')\phi \left(\frac{1}{(1+\delta')^2}(T-4.5\kappa_f)\right) &\geq \\
\phi(T-5\kappa_f)
\end{aligned}$$

where the first inequality follows by Fact 5.2, and the second and third by the taking a sufficiently small δ'. Since $T \geq \frac{5\kappa_f}{\delta}$, we have that $(1-\delta)T \geq T - 5\kappa_f$, thus the number of fruits in the sequence is at least

$$(1-\delta)\phi T$$

References

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In *CRYPTO'05*, 2005.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *CRYPTO'92*, pages 139–147, 1992.

[ES14] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Advances in Cryptology-EUROCRYPT 2015*, pages 281–310. Springer, 2015.

[HP15] Joseph Y. Halpern and Rafael Pass. Algorithmic rationality: Game theory with costly computation. *J. Economic Theory*, 156:246–268, 2015.

[KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols, 2015.

[LSZ15] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Financial Crypto'15*, 2015.

[mtg10] mtgox. `https://bitcointalk.org/index.php?topic=2227.msg29606#msg29606`, 2010.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[PSS16] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks, 2016.

[PS16] Rafael Pass and Elaine Shi. Hybrid consensus, 2016.

[SSZ16] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Crypto'16*, 2016.

[SZ15] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

A Appendix

We recall the standard Chernoff bound.

Lemma A.1 (Multiplicative Chernoff Bound)**.** *Let $X_1, \ldots X_n$ be independent Boolean random variables, such that for all i, $\Pr[X_i = 1] = p$; let X be the sum of these variables, and μ be the expectation of the sum. Then for any $\delta \in (0, 1]$, we have*

$$Pr\left[X > (1+\delta)\,\mu\right] < e^{-\Omega\left(\delta^2 \mu\right)}$$

$$Pr\left[X < (1-\delta)\,\mu\right] < e^{-\Omega\left(\delta^2 \mu\right)}$$