

On Bitcoin Cash’s Target Recalculation Functions

Juan Garay¹ and Yu Shen¹

¹Texas A&M University, {garay, shenyu.tcv}@tamu.edu

June 3, 2021

Abstract

Bitcoin Cash, created in 2017, is a “hard fork” from Bitcoin responding to the need for allowing a higher transaction volume. This is achieved by a larger block size, as well as a new difficulty adjustment (target recalculation) function that acts more frequently (as opposed to Bitcoin’s difficulty adjustment happening about every two weeks), resulting in a potentially different target *for each block*. While seemingly achieving its goal in practice, to our knowledge there is no formal analysis to back this proposal up.

In this paper we provide the first formal cryptographic analysis of Bitcoin Cash’s target recalculation functions—both ASERT and SMA (current and former recalculation functions, respectively)—against all possible adversaries. The main distinction with respect to Bitcoin’s is that they are no longer epoch-based, and as such previous analyses fail to hold. We overcome this technical obstacle by introducing a new set of analytical tools focusing on the “calibration” of blocks’ timestamps in sliding windows, which yield a measure of closeness to the initial block generation rate. With that measure, we then follow the analytical approach developed in the *Bitcoin backbone protocol* [Eurocrypt 2015 and follow-ups] to first establish the basic properties of the blockchain data structure, from which the properties of a robust transaction ledger (namely, *Consistency* and *Liveness*) can be derived.

We compare our analytical results with data from the Bitcoin Cash network, and conclude that in order to satisfy security (namely, properties satisfied except with negligible probability in the security parameter) considerably larger parameter values should be used with respect to the ones used in practice.

Contents

1	Introduction	3
2	Preliminaries	4
3	Bitcoin Cash	6
3.1	The ASERT Target Recalculation Function	6
3.2	ASERT: Preliminary Analysis	7
4	Full Security Analysis	13
4.1	The Bitcoin Cash Backbone Protocol	13
4.2	ASERT: Full Analysis	16
5	The SMA Target Recalculation Function	23
5.1	SMA: Analysis Overview	24
6	Comparison with the Bitcoin Cash Network	25
6.1	The Bitcoin Cash Network	25
6.2	Conclusions	27
	References	28
A	Mathematical Facts	29
B	Preliminaries (cont'd)	30
B.1	Model	30
B.2	Blockchain Notation	33
B.3	Blockchain Properties	33
B.4	Ledger Properties	34
C	Proof That All Executions Are Typical	34
D	The SMA Target Recalculation Function (cont'd)	37
D.1	SMA: Full Analysis	37

1 Introduction

While opening up a new era in the area of cryptocurrencies, Nakamoto’s Bitcoin protocol [Nak09b, Nak09a] has been criticized for its heavy use of the computational resources needed by its underlying proof of work (PoW) mechanism as well as its relatively long settlement time of transactions. As a consequence, a number of alternative cryptocurrencies have been proposed with the purpose of ameliorating the above issues. One such proposal is Bitcoin Cash (BCH)¹, created in August 2017 as a “hard fork” of Bitcoin, with the original motivation of increasing the size of blocks, and thus allowing more transactions to be processed.

Due to lesser prominence and popularity, the computational “investment” on these alternate cryptocurrencies is relatively low (for example, the hashing power invested on Bitcoin Cash is approximately 5% of that on Bitcoin). Moreover, miners are able to evaluate their expected reward and rapidly switch among different blockchains in order to achieve a higher profit, giving rise to an environment where the number of participating miners on these minor blockchains may fluctuate wildly, which in turn has a direct effect on suitable difficulty (of PoWs’) recalculation mechanisms².

The above two aspects—desired higher transaction throughput and higher participation variation—are the motivation for this work. We focus on Bitcoin Cash as a representative of a newly proposed target recalculation function, and perform a formal analysis of the protocol’s security under such dynamic environment. The importance of an adequate target recalculation mechanism has already been pointed out in [GKL17], where it is observed that if it is removed, the blockchain protocol becomes insecure in the dynamic setting *even* if all parties behave honestly, resulting in a blockchain that will diverge substantially (i.e., spawning “forks”) as the number of miners keeps increasing, thus becoming vulnerable to many known cryptographic attacks. Furthermore, an inadequate target recalculation function may break the balance between miners’ invested hashing power and reward, thus reducing their confidence in the system and leading them to quit the mining pool, arguably a situation that should be avoided.

Bitcoin Cash’s target recalculation algorithm has gone through three stages. When created, the recalculation mechanism was a combination of Bitcoin’s target recalculation function and an *emergency difficulty adjustment* (EDA) mechanism, which would suddenly enlarge targets by 25% (i.e., decrease mining difficulty by 20%) if the block generating interval of 6 blocks exceeds 12 hours.

In November 2017, this initial function was replaced by a new function called *SMA* (for *Simple Moving Average*, or “cw-144”). At a high level, SMA is analogous to Bitcoin’s recalculation function in the sense that it determines the next target based on an “epoch” of blocks, except that in the new algorithm the target value is recalculated more frequently—in fact, the target for *each block* varies. Moreover, the epoch of blocks changes with every block in a “sliding window” fashion. Generally speaking, the SMA function calculates target for the next block based on the length of the epoch and the *average* target of the blocks in the epoch.

Finally, the recent (November 2020) update introduces a control-theory-inspired recalculation function called *ASERT* (for *Absolutely Scheduled Exponentially Rising Targets*) [WISK20]), which is completely different from previous recalculation functions. ASERT is not epoch-based, and adjusts the difficulty level of each block simply based on its timestamp and height difference with the “anchor” block. Specifically, once an anchor block is chosen, a timestamp can be computed for all the subsequent blocks according to its height and ideal block generating interval. The difficult adjustment is an exponential function of the block’s timestamp deviation from its scheduled

¹<https://www.bitcoincash.org/>.

²As a reference, Bitcoin adjusts the PoWs’ difficulty level every 2016 blocks – approximately every two weeks.

timestamp. the target changes is controlled by a “smoothing factor” m , which we will show is a crucial parameter in our analysis.

Overview of our results. Our main contribution is establishing under what conditions regarding party fluctuation Bitcoin Cash’s target recalculation functions (both ASERT and SMA) achieve a steady and close to the ideal block generation rate, given that they are not epoch-based, and the recalculation mechanism is invoked for every block—and further, in the case of ASERT, that it is “memory-less,” meaning that the target for one block is only decided by the current timestamp and the block’s height. As such, previous analyses based on the duration of an epoch no longer hold, and new analytical tools are needed.

As in prior work on dynamic environments, bounds on the ways that miners come in and drop out of the computation are necessary for things to work. We suggest a new methodology to capture how the number of parties can fluctuate. In a nutshell, our definition is comprised of two parts concerning *both* short-term and long-term participation. In such context, we first (Section 3) perform a preliminary analysis of the ASERT function to establish whether in a suitable respecting environment, the blocks in chains created according to the function will have blocks with timestamps to make the probability of producing the next blocks close to the ideal block generation rate (we will call such timestamps “good”). Through a closeness measure based on “calibrated timestamps” and probabilistic analysis we conclude that they do.

Our conclusions then serve as a crucial part of the complete security analysis (Section 4), where, following [GKL15, GKL17], we present an abstraction of the protocol we term the *Bitcoin Cash backbone*, and follow the “template” of establishing two main properties of the blockchain data structure—“common prefix” and “chain quality”—which serve as an intermediate step to build a robust transaction ledger. As a result, (our abstraction of) the Bitcoin Cash protocol with chains of variable difficult using the ASERT function running in a bounded-delay network and suitably parameterized, satisfies, with overwhelming probability, the properties of consistency and liveness.

In addition (Section 5), we also provide a description and analysis of Bitcoin Cash’s previous target recalculation function SMA. Even though the function has now been deprecated, it provides insights and elements of comparison against ASERT with regard to party fluctuation.

Finally (Section 6), we compare our results with data from the Bitcoin Cash network, from which we extract the actual party fluctuation rate and network delay. Our main conclusion is that in order to satisfy security (namely, properties satisfied except with negligible probability) increased parameter values should be used with respect to the ones used in practice—specifically, a larger value of m (the smoothing factor in ASERT and epoch length in SMA)—concretely, $m = 432$, compared to the value $m = 288$ that is being used in ASERT (which in turn corresponds to 2 days)—should be adopted. In addition, regarding the SMA function, a larger dampening filter $\tau = 8$ in the SMA function should be used, instead of $\tau = 2$, which is the value that was last used. Lastly, our comparison with the existing Bitcoin Cash network shows that the ASERT function performs better than the SMA function under a pronounced party fluctuation.

Due to space limitations, some of the proofs and detailed protocol descriptions, and other complementary material are presented in the appendix.

2 Preliminaries

In this section we present the network model where we analyze Bitcoin Cash’s target recalculation functions as well as the Bitcoin Cash protocol abstraction, as well as some basic blockchain notation. These notions and terminology follow closely [GKL17, GKL20], and therefore the presentation

here is succinct, except for the extension of the notion of respecting environments with respect to [GKL20]. More formal details about the model are presented in Appendix B.1.

Model. We describe our protocol in the bounded-delay (aka “partially synchronous”) network model considered in [PSS17, GKL20], where there is an upper bound Δ in the delay (measured in number of rounds) that the adversary may inflict to the delivery of any message. The precise value of Δ will be unknown to the protocol. “Rounds” still exist in the model, but now these are not synchronization rounds where messages are supposed to be delivered to honest parties (“miners”). At any given time (round), a fraction of the parties may be “corrupted” and controlled by an adversary \mathcal{A} , directing them to behave in an arbitrary and potentially malicious manner. The underlying communication graph is not fully connected and messages are delivered through a “diffusion” mechanism.

As in [GKL20], we assume a dual hash function/network functionality that is available to all parties running the protocol and the adversary, and which abstracts the access of the parties to the hash function and the network. The hash function aspect relates to the parties attempting to solve “proofs of work” (PoW) [DN92] during the execution of the protocol, and is modeled as parties having access to the oracle $H(\cdot)$. In our analysis, each honest party P_i is allowed to ask one query to the oracle in each round, but unlimited queries for verification. The adversary \mathcal{A} , on the other hand, is given at each round r a number of queries that cannot exceed t_r . The “diffusion” aspect of the communication allows the order of messages to be controlled by \mathcal{A} . Furthermore, the adversary is allowed to spoof the source information on every message (i.e., communication is not authenticated).

Garay *et al.* [GKL20] refer to the setting defined by the two series $\mathbf{n} = \{n_r\}_{r \in \mathbb{N}}$ and $\mathbf{t} = \{t_r\}_{r \in \mathbb{N}}$, representing the number of “ready” honest parties and the bound on corrupted parties that may be activated at each round r , with the above bounded access to the random oracle functionality, as the *dynamic setting*. As pointed out in [GKL17], protocol’s properties cannot be satisfied under arbitrary sequences of parties, and restrictions are in order. In our model, one important difference with respect to [GKL17, GKL20] is the bound on the *short-term* variation of the number of parties, in addition to the long-term bound, which results in the following definition of accepted variation terms on the number of parties³:

Definition 1. For $\gamma, \Gamma \in \mathbb{R}^+$, we call a sequence $(n_r)_{r \in \mathbb{N}}$ $(\langle \gamma, \sigma \rangle, \langle \Gamma, \Sigma \rangle)$ -*respecting* if it holds that in a sequence of rounds S with $|S| \leq \Sigma$ rounds, $\max_{r \in S} n_r \leq \Gamma \cdot \min_{r \in S} n_r$ and for any consecutive sub-sequence rounds $S' \preccurlyeq S$ with $|S'| \leq \sigma$ rounds, $\max_{r \in S'} n_r \leq \gamma \cdot \min_{r \in S'} n_r$.

The term $\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}$ denotes the random variable ensemble, when considering a “standalone” execution without any auxiliary information as well as concatenation of view of all parties ever activated. In our theorems we will be concerned with *properties* of protocols Π . Such properties will be defined as predicates over $\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}$ with a small probability of error in κ as well as in a parameter k that is selected from $\{1, \dots, \kappa\}$.

Blockchain notation. A *block* is a quadruple of the form $B = \langle r, st, x, ctr \rangle$ with $st \in \{0, 1\}^\kappa$, $x \in \{0, 1\}^*$ and $r, ctr \in \mathbb{N}$. They satisfy the predicate $\text{validblock}_q^T(B)$ defined as $(H(ctr, G(r, st, x)) < T) \wedge (ctr \leq q)$ where $G(\cdot), H(\cdot)$ are cryptographic hash functions with output in $\{0, 1\}^\kappa$. A *blockchain* is a sequence of *blocks*. The rightmost block is the *head* of the block denoted $\text{head}(\mathcal{C})$. Note that the empty string ε is also a chain and $\text{head}(\varepsilon) = \varepsilon$. A chain \mathcal{C} with $\text{head}(\mathcal{C})$ can be extended by

³Definition 1 is consistent with the notion introduced in [GKL17], except that there the party fluctuation is expressed with respect to the number of honest parties. Note that the two fluctuations are related by a constant (concretely, $2 - \delta$; see Table 1). As it turns out, expressing statements in terms of honest-party fluctuation will considerably simplify them.

appending a valid block $B = \langle r', st', x', ctr' \rangle$ that satisfies $(st' = H(ctr, G(r, st, x))) \wedge (r' > r)$. By convention, any valid block may extend the chain $\mathcal{C} = \varepsilon$. The *length* of a chain $\text{len}(\mathcal{C})$ is its number of blocks. Consider a chain \mathcal{C} of length ℓ and any nonnegative integer k , we denote by $\mathcal{C}^{\lceil k}$ the chain resulting from “pruning” the k rightmost blocks. Note that for $k \geq \text{len}(\mathcal{C})$, $\mathcal{C}^{\lceil k} = \varepsilon$. If \mathcal{C}_1 is a prefix of \mathcal{C}_2 we write $\mathcal{C}_1 \preceq \mathcal{C}_2$.

3 Bitcoin Cash

A salient distinction of Bitcoin Cash (in addition to the size of blocks) with respect to Bitcoin is its target recalculation function—ASERT. Intriguingly, and in contrast to SMA, this function is “memory-less,” so one thing we would like to find out right away is whether such a function is able to maintain, under the conditions allowed by our model, a steady block generation rate. If that’s the case, we will then proceed with the protocol abstraction and analysis to establish under what values of parameters the blockchain and ledger properties (cf. Section 4) can be satisfied. (The latter is also required for the SMA analysis.)

3.1 The ASERT Target Recalculation Function

The November 2020 Bitcoin Cash update introduced a new difficulty adjustment algorithm, called *ASERT* (for “Absolutely Scheduled Exponentially Rising Targets”), aimed at achieving a stable block generation interval and transaction confirmation time as well as reducing the advantage of non-steady mining strategies. The new algorithm is derived from the control theory literature, and, specifically, from the notion of *Exponentially Moving Average* (EMA), a type of moving average that places greater weight on the most recent data points, in contrast, to *Simple Moving Average* (SMA, the previous target recalculation function), which applies an equal weight to all the observations in the period. For additional details, refer to [WISK20], where a mathematical derivation of this function based on exponential smoothing, a common technique for removing noise from time series data, is provided.

ASERT adjusts the target based on the *anchor block*, i.e., a block whose target is denoted by T_0 , and is used as a reference for all subsequent blocks. We use $f \in (0, 1)$ to denote the ideal block generation rate (and thus $1/f$ represents the ideal block generation interval). At a high level, for a given block, ASERT compares its timestamp with the scheduled timestamp, which is a product of the ideal block generating interval and the block’s *height* difference (e.g., for the i -th block, it is $(i - 1)/f$). If the block’s timestamp is ahead of the scheduled time, which means that the number of miners is larger than that corresponding to the anchor block, ASERT decreases the target (i.e., raises the difficulty of generating a block); if it falls behind the scheduled time, then the target is increased (i.e., the difficulty is reduced).

The amount by which the target is changed is based on a value m , the *decay* or *smoothing* factor. Specifically, the target is adjusted exponentially based on the ratio of time difference and the smoothing time (i.e., m/f). For example, if the smoothing time is 2 days, then a block with a timestamp 2 days ahead of the scheduled timestamp would have a target whose value is half of the anchor target’s.

Formally, ASERT is defined as follows. (Note that we assume the anchor block to be the first block and timestamps start at 0.)

Definition 2 (ASERT). For fixed constants m, T_0 , the target calculation function $D_{\text{ASERT}} : \mathbb{Z}^* \rightarrow \mathbb{R}$ is defined as

$$D_{\text{ASERT}}(\varepsilon) = T_0 \text{ and } D_{\text{ASERT}}(r_1, \dots, r_v) = T_0 \cdot 2^{(r_v - (v-1)/f)/(m/f)}, \quad (1)$$

where (r_1, \dots, r_v) corresponds to a chain of v blocks with r_i the timestamp of the i -th block.

Note that, as opposed to Bitcoin’s target recalculation algorithm [GKL17] and also to Bitcoin Cash’s recalculation function (SMA, Section 5), ASERT is not epoch based. Moreover, and as mentioned earlier, ASERT is *memoryless*—i.e., the target for one block is only decided by the current timestamp and the block’s height. No matter what timestamps the previous blocks have, they would not influence the current block’s target value.

Removing the function’s “dampening” filter raises the question of whether it would suffer from Bahack’s raising difficulty attack [Bah13]. It turns out that it does not, since Equation (1) intrinsically prevents the difficulty from a sudden sharp increase. Concretely, assuming monotonically increasing timestamps, even if the adversary produces m blocks with the same timestamp, he can only double the difficulty value.

3.2 ASERT: Preliminary Analysis

We now provide a preliminary analysis of the ASERT function to establish whether in a suitable environment, the blocks in chains created according to the function will have blocks with timestamps to make the probability of producing the next blocks close to the ideal block generating rate. (With foresight, we will call such timestamps “good.”) Our preliminary conclusions will then serve as a crucial part of the complete security analysis, after we introduce the Bitcoin Cash protocol abstraction.

Our security parameter is hash function length κ , and we let the smoothing factor $m = \text{polylog}(\kappa)$. Our probability space is over all executions of length at most some polynomial in κ . We will denote by \mathbf{Pr} the probability measure of this space.

We use n_r to denote the total number of parties at round r ([GKL17, GKL20] uses n_r to denote the number of *honest* parties), and t_r to denote the number of corrupted parties. Thus, the number of honest parties at round r is $n_r - t_r$. For simplicity, we use $h_r = n_r - t_r$. This follows the tradition in the secure multiparty computation literature and the notation in [GKL15].

Recall that in our model, each party’s query bound to the random oracle (RO) per round is $q = 1$. Now suppose that at round r exactly n parties query the RO with a target T . Then the probability that at least one of them will succeed is

$$f(T, h) = 1 - (1 - pT)^h \leq phT, \text{ where } p = 1/2^\kappa.$$

We let $f_0 = f(T_0, h_0)$, where T_0 and h_0 are the initial target and estimate of number of honest parties, respectively. The objective of the target recalculation mechanism is to maintain a target T for each party such that $f(T, h_r) \approx f_0$ for all rounds r . For notational simplicity, we will drop the subscript from f_0 , and will always specify the two arguments of $f(\cdot, \cdot)$ to avoid confusion.

We say round r is a target-recalculation point of a valid chain \mathcal{C} , if there is a block with timestamp r . Recall that our goal is to show that all the target recalculation points on a chain using the ASERT function are close to f , i.e., “good.” How close should this generation rate (ph_rT) be?

Intuitively, the block generation rate should satisfy $f/\Gamma \leq ph_rT \leq \Gamma f$, when considered in a respecting environment with long-term party fluctuation ratio Γ (recall Definition 1). Moreover, adversarial parties can choose to keep silent, hence decelerate the block production process. One might consider adding an honest-party advantage, say, δ to the lower bound—i.e., $f/(2 - \delta)\Gamma \leq ph_rT \leq \Gamma$. As we shall see, it turns out that this modification is not sufficient for a satisfactory security analysis, and is inadequate to absorb all the errors in our model. I.e., the adversary would be able to wait for an appropriate moment to act and then disturb the regular block production.

Consequently, we need a looser bound to compensate for adversarial behavior as well as for errors introduced by the (peer-to-peer and dynamic) network. Taking this into account, we proceed to define a “good” target-recalculation point:

Definition 3 (Goodness). A target-recalculation point r is *good* if the target T for the next block satisfies $f/2(2 - \delta)\Gamma^3 \leq ph_r T \leq 2\Gamma^3 f$.

As mentioned above, ASERT not being epoch-based means that previous analyses regarding the “goodness” of recalculation points for Bitcoin [GKL20] (as well as for SMA—Section 5) do not hold.

We start our analysis by presenting some basic observations regarding the ASERT function. Note that the target for the next block is merely related to the block’s timestamp and height. For the i -th block with timestamp r and corresponding number of honest parties h_r , it is not hard to see that if $r = (i - 1)/f + (m/f) \log(h_0/h_r)$, the i -th block would have block generation rate exactly f . We call this timestamp r the *calibrated timestamp* for block B_i . In addition, r is a good target recalculation point if it satisfies

$$\frac{i - 1}{f} + \frac{m}{f} \log(2(2 - \delta)\Gamma^3 \cdot \frac{h_0}{h_r}) \leq r \leq \frac{i - 1}{f} + \frac{m}{f} \log(2\Gamma^3 \cdot \frac{h_0}{h_r}). \quad (2)$$

We now define a new random variable to describe the deviation of timestamps: We use X_i to express by how much the i -th block deviates from its calibrated timestamp. For the i -th block with timestamp r_i and number of honest parties h_i ,

$$X_1 = 0 \text{ and } X_{i+1} = X_i + (r_{i+1} - r_i) - \frac{1}{f} - \frac{m}{f} \log \frac{h_{i+1}}{h_i} \text{ for } i \geq 0.$$

The three parts in the definition of X_{i+1} are as follows: (1) $(r_{i+1} - r_i)$ represents the difference between their timestamps, (2) $1/f$ is the ideal block interval, and (3) $(m/f) \log(h_{i+1}/h_i)$ is the difference between the respective number of honest parties. For “good” blocks, variable X_i should satisfy $-(m/f) \log 2(2 - \delta)\Gamma^3 \leq X_i \leq (m/f) \log 2\Gamma^3$.

As defined, X_i is sensitive to the fluctuation of number of parties. Since we can only bound the fluctuation rate during a fixed number of rounds, X_i is not suitable for the analysis. To overcome this, we consider a new random variable W_i within a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment, and then show that if this new random variable satisfies certain conditions, then X_i presented above satisfies the ideal bound (2).

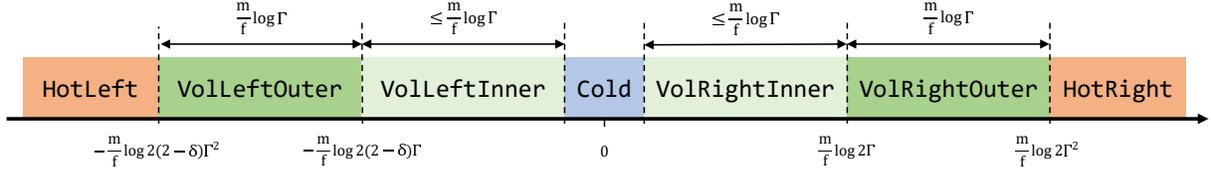
In more detail, we consider the calibrated timestamp $r = (i - 1)/f + (m/f) \log(h_0/h_r)$, and a sliding window of $4(m/f) \log \Gamma$ rounds starting with block B_u and number of honest parties h_u . For each subsequent block in this window, we replace h_r with h_u and call $r' = (i - 1)/f + (m/f) \log(h_0/h_u)$ the *relatively calibrated timestamp with respect to B_u* for i -th block $B_i, i \geq u$. We can now define a new random variable W_i expressing by how much the i -th block deviates from its relatively calibrated timestamp (wrt B_u). That is, for i -th block with timestamp r_i ,

$$W_u = X_u \text{ and } W_{i+1} = W_i + (r_{i+1} - r_i) - \frac{1}{f} \text{ for } i \geq u.$$

Now the definition of the random variable only consists of two parts: The difference between their timestamps, and the ideal block interval. For good target recalculation points, W_i should satisfy

$$-\frac{m}{f} \log 2(2 - \delta)\Gamma^2 \leq W_i \leq \frac{m}{f} \log 2\Gamma^2.$$

Figure 1: The states based on the values of random variable W_i .



Next, we define seven states based on values of the random variable W_i . Studying the possible transitions between them allows us to establish that target recalculation points are good. Refer to Figure 1. Let h_{\max}, h_{\min} denote the maximum and minimum number of parties during the sliding window, respectively.

$$\begin{aligned}
 \text{HotLeft}_i &\triangleq W_i < -\frac{m}{f} \log 2(2 - \delta)\Gamma^2 \\
 \text{VolatileLeftOuter}_i &\triangleq -\frac{m}{f} \log 2(2 - \delta)\Gamma^2 \leq W_i < -\frac{m}{f} \log 2(2 - \delta)\Gamma \\
 \text{VolatileLeftInner}_i &\triangleq -\frac{m}{f} \log 2(2 - \delta)\Gamma \leq W_i < -\frac{m}{f} \log \frac{2(2 - \delta)h_{\max}}{h_u} \\
 \text{Cold}_i &\triangleq -\frac{m}{f} \log \frac{2(2 - \delta)h_{\max}}{h_u} \leq W_i \leq \frac{m}{f} \log \frac{2h_u}{h_{\min}} \\
 \text{VolatileRightInner}_i &\triangleq \frac{m}{f} \log \frac{2h_u}{h_{\min}} < W_i \leq \frac{m}{f} \log 2\Gamma \\
 \text{VolatileRightOuter}_i &\triangleq \frac{m}{f} \log 2\Gamma < W_i \leq \frac{m}{f} \log 2\Gamma^2 \\
 \text{HotRight}_i &\triangleq W_i > \frac{m}{f} \log 2\Gamma^2
 \end{aligned}$$

States VolatileLeftOuter and $\text{VolatileRightOuter}$ are of fixed length $(m/f) \log \Gamma$, while states VolatileLeftInner and $\text{VolatileRightInner}$ are of length at most $(m/f) \log \Gamma$. These lengths will play a significant role in the following analysis of the ASERT function.

We aim to show that for blocks B_u, \dots, B_v generated in an interval of length $4(m/f) \log \Gamma$ rounds, the following holds:

- For a block B_i , $i > u$ with W_i (w.r.t. B_u) in state Cold , we can construct a new $4(m/f) \log \Gamma$ -round sliding window with W_i (w.r.t. B_i) in state VolatileLeftInner , $\text{VolatileRightInner}$ or Cold .
- If W_u is in state VolatileLeftInner , $\text{VolatileRightInner}$ or Cold , the probability of W_i , $i > u$ reaching HotLeft or HotRight is negligible.
- If W_u is in state VolatileLeftInner , $\text{VolatileRightInner}$ or Cold , W_i , $i > u$ will return to Cold with overwhelming probability.

Lemma 1 below follows from the definition of each state and party fluctuation; Lemma 2 establishes a basic property the volatile states satisfy.

Lemma 1. *For a block B_v , if W_v w.r.t. B_u is in state Cold , then W_v w.r.t. B_v is in state VolatileLeftInner , $\text{VolatileRightInner}$ or Cold .*

Proof. While w.r.t. B_u , $X_v = W_v + (m/f) \log(h_v/h_u)$. Combine it with Cold_v , as well as $h_{\max} \leq \Gamma \cdot h_v, h_v \leq \Gamma \cdot h_{\min}$, we get

$$\begin{aligned} -\frac{m}{f} \log 2(2-\delta)\Gamma &\leq -\frac{m}{f} \log \frac{2(2-\delta)h_{\max}}{h_u} + \frac{m}{f} \log \frac{h_v}{h_u} \\ &\leq X_v = W_v + \frac{m}{f} \log \frac{h_v}{h_u} \leq \frac{m}{f} \log \frac{2h_u}{h_{\min}} + \frac{m}{f} \log \frac{h_v}{h_u} \leq \frac{m}{f} \log 2\Gamma. \end{aligned}$$

By the definition of W_v w.r.t. B_v , $W_v = X_v$, therefore in state VolatileLeftInner , $\text{VolatileRightInner}$ or Cold . \square

Lemma 2. *Until the next block is produced, if W_i is in the state VolatileLeftOuter or VolatileLeftInner , the block generation rate is always below $f/2$; if W_i is in the state $\text{VolatileRightInner}$ or $\text{VolatileRightOuter}$, the block generation rate is always above $2f$.*

Proof. For the first part, our goal is to show that even if the adversary and the honest parties join force, they cannot achieve the block generation rate over $f/2$. Suppose i -th block has timestamp r and number of honest parties h_r . If $W_i < -(m/f) \log[2(2-\delta)h_{\max}/h_u]$, the target of B_i satisfies

$$T_r < T_0 \cdot 2^{(\frac{i-1}{f} + \frac{m}{f} \log \frac{h_0}{h_u} - \frac{m}{f} \log \frac{2(2-\delta)h_{\max}}{h_u} - \frac{i-1}{f})/(m/f)} = \frac{T_0}{2} \cdot \frac{h_0}{(2-\delta)h_{\max}} \leq \frac{T_0}{2} \cdot \frac{h_0}{(2-\delta)h_r}.$$

Therefore, the block generating rate at round r is $pT_r h_r < f/[2(2-\delta)]$. Note that $pT_r h_{\max} < f/[2(2-\delta)]$ as well, which implies that while the number of honest parties may raise during the rounds till the next block will be produced, the block generating rate will never exceed $f/[2(2-\delta)]$. Moreover, the adversary may join force to accelerate the block production. Recall that $\forall r, t_r \leq (1-\delta)h_r$, after the adversary joins, the block production rate is still below $f/2$.

For the second part, we prove that the block generating rate will not fall below $2f$ when the honest parties keep solely working. Similarly, assuming i -th block has timestamp r and number of honest parties h_r as well as $W_i \geq (m/f) \log(2h_u/h_{\min})$. Thus, the corresponding target

$$T_r > T_0 \cdot 2^{(\frac{i-1}{f} + \frac{m}{f} \log \frac{h_0}{h_u} + \frac{m}{f} \log \frac{2h_u}{h_{\min}} - \frac{i-1}{f})/(m/f)} = 2T_0 \cdot \frac{h_0}{h_{\min}} \geq 2T_0 \cdot \frac{h_0}{h_r}.$$

By $pT_r h_{\min} > 2f$, we learn that the block generating rate is always above $2f$. \square

We are now ready to establish the probability of “escaping” from a volatile state to a hot state.

Lemma 3. *Consider blocks B_u, \dots, B_v with timestamps $r_v - r_u \leq 4(m/f) \log \Gamma$ in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment. If W_u is in state VolatileLeftInner , $\text{VolatileRightInner}$ or Cold , the probability of W_i , $i > u$ reaching HotLeft or HotRight is negligible.*

Proof. Regarding the probability of escaping from VolatileLeftOuter , by Lemma 2, at every round it will succeed producing a block with probability at most $f/2$. We view the number of blocks produced as a binomial distribution with success probability $f/2$. And, for worst case, W_i begins at the leftmost point in $\text{VolatileRightInner}$, it has to go leftwards for $(m/f) \log \Gamma$ in order to reach HotLeft .

Since $W_{i+1} = W_i + (r_{i+1} - r_i) - 1/f$, we get $W_v = W_u + r_v - r_u - (u - v)/f$. Assume now $W_u = -(m/f) \log 2(2-\delta)\Gamma$, if W_v is in HotLeft , $r_v - r_u - (u - v)/f < -(m/f) \log \Gamma$. Obviously, this will never happen in the first $(m/f) \log \Gamma$ rounds. For the rounds with index in $\{(m/f) \log \Gamma + 1, \dots, 4(m/f) \log \Gamma\}$, split them into segments with length $1/f$. For r in i -th segment with index $\{(m/f) \log \Gamma + (i-1)/f + 1, \dots, (m/f) \log \Gamma + i/f\}$, suppose we produce a block B_v , in expectation,

parties will succeed for $\lfloor (m \log \Gamma + i - 1)/2 \rfloor$ times in these rounds. If they succeed for more than $(m \log \Gamma + i)$ times, then $r_v - r_u - (u - v)/f < -(m/f) \log \Gamma$, thus reach **HotLeft**.

Note that $m \log \Gamma + i \geq 2 \cdot \lfloor (m \log \Gamma + i - 1)/2 \rfloor$ always holds. By Theorem 19, let Z_i, \dots, Z_T ($T = r$) be independent variables with $E[Z_i] = f/2$ and $Z_i \in \{0, 1\}$. Let $Z = \sum_{i=1}^T Z_i$, $\mu = \sum_{i=1}^T f/2 = E[Z] = \lfloor (m \log \Gamma + i - 1)/2 \rfloor \geq m \log \Gamma$. Then, for $\Lambda = 1$, we get

$$\Pr[Z \geq (1 + \Lambda)\mu] \leq \exp \left[-\frac{\Lambda^2}{2 + \Lambda} \cdot m \log \Gamma \right] \leq 2^{-\Omega(m)}.$$

Eventually, this may happen for $3(m/f) \log \Gamma$ times, thus we get the negligible escape probability

$$1 - (1 - 2^{-\Omega(m)})^{3\frac{m}{f} \log \Gamma} \geq 3\frac{m}{f} \log \Gamma \cdot 2^{-\Omega(m)} = 2^{-\Omega(\text{polylog}(\kappa))}.$$

Consider the the probability of escaping from **VolatileRightOuter**, by Lemma 2, at every round it will succeed producing a block with probability at least $2f$. We view the number of blocks produced as a binomial distribution with success probability $2f$. And, for worst case, W_i begins at the rightmost point in **VolatileRightInner**, it has to go rightwards for $(m/f) \log \Gamma$ in order to reach **HotRight**.

Since $W_{i+1} = W_i + (r_{i+1} - r_i) - 1/f$, we get $W_v = W_u + r_v - r_u - (u - v)/f$. Assume now $W_u = (m/f) \log 2\Gamma$, if W_v is in **HotRight**, $r_v - r_u - (u - v)/f > (m/f) \log \Gamma$. Obviously, this will never happen in the first $(m/f) \log \Gamma$ rounds. For the rounds with index in $\{(m/f) \log \Gamma + 1, \dots, 4(m/f) \log \Gamma\}$, split them into segments with length $1/f$. For r in i -th segment with index $\{(m/f) \log \Gamma + (i - 1)/f + 1, \dots, (m/f) \log \Gamma + i/f\}$ suppose we produce a block B_v , in expectation, parties will succeed for $2(m \log \Gamma + i - 1)$ times in these rounds. If they succeed for less than i times, then $r_v - r_u - (u - v)/f > (m/f) \log \Gamma$, thus reach **HotRight**.

Note that $i \leq (1/2) \cdot 2(m \log \Gamma + i - 1)$ always holds. By Theorem 19, let Z_i, \dots, Z_T ($T = r$) be independent variables with $E[Z_i] = 2f$ and $Z_i \in \{0, 1\}$. Let $Z = \sum_{i=1}^T Z_i$, $\mu = \sum_{i=1}^T 2f = E[Z] = 2(m \log \Gamma + i - 1) \geq 2 \log \Gamma$. Then, for $\Lambda = 1/2$, we get

$$\Pr[Z \leq (1 - \Lambda)\mu] \leq \exp \left[-\frac{\Lambda^2}{2 + \Lambda} \cdot 2m \log \Gamma \right] \leq 2^{-\Omega(m)}.$$

Eventually, this may happen for $3(m/f) \log \Gamma$ times, and the final escape probability

$$1 - (1 - 2^{-\Omega(m)})^{3\frac{m}{f} \log \Gamma} \geq 3\frac{m}{f} \log \Gamma \cdot 2^{-\Omega(m)} = 2^{-\Omega(\text{polylog}(\kappa))}$$

is still negligible. □

Next, we focus on the “return” probability. Since W_i will travel far away from the *relatively calibrated timestamp* (i.e., **HotLeft** or **HotRight**) only with negligible probability, and once it reaches **Cold** we are done, we consider the following two bad events:

- During $4(m/f) \log \Gamma$ rounds, beginning at $-(m/f) \log 2(2 - \delta)\Gamma$ (the leftmost point of **VolatileLeftInner**), W_i stays in state **VolatileLeftOuter** and **VolatileLeftInner**.
- During $4(m/f) \log \Gamma$ rounds, beginning at $(m/f) \log 2\Gamma$ (the rightmost point of **VolatileRightInner**), W_i stays in the state **VolatileRightInner** and **VolatileRightOuter**.

We show that by the concentration of the binomial distribution, these two bad events happen only with negligible probability. Note that our results are achieved considering the largest distance W_i needs to travel and with worst success probability. For those events that start closer to the relatively calibrated timestamp, the events’ probability will be much lower.

Lemma 4. Consider blocks B_u, \dots, B_v with $r_v - r_u \leq 4(m/f) \log \Gamma$ in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment. If W_u is in state `VolatileLeftInner`, `VolatileRightInner` or `Cold`, W_i , $i > u$ will return to `Cold` with overwhelming probability.

Proof. We consider the two bad events that makes W_i fail to return `Cold`.

For the first event of staying in the left-side states, Lemma 2 shows that at every round it will succeed producing a block with probability at most $f/2$. We view the number of blocks produced as a binomial distribution with success probability $f/2$. And, since we assume it begins at the leftmost point in `VolatileLeftInner`, it has to go rightwards for at most $(m/f) \log \Gamma$ in order to reach `Cold`.

Consider the first $(4m/f) \log \Gamma$ rounds with blocks $\{B_u, \dots, B_v\}$. Since $W_{i+1} = W_i + (r_{i+1} - r_i) - 1/f$, we get $W_v = W_u + r_v - r_u - (u - v)/f$. Assume now $W_u = -(m/f) \log 2(2 - \delta)\Gamma$, if W_v is in `Cold`, $r_v - r_u - (u - v)/f > (m/f) \log \Gamma$. In expectation, parties will succeed for $(2m \log \Gamma)$ times in $(4m/f) \log \Gamma$ rounds. If they succeed for more than $(3m \log \Gamma)$ times, it cannot satisfy $r_v - r_u - (u - v)/f > (m/f) \log \Gamma$, thus fails to reach `Cold`, i.e., W_i still falls in `VolatileLeftInner`.

By Theorem 19, let Z_i, \dots, Z_T ($T = (4m/f) \log \Gamma$) be independent variables with $E[Z_i] = f/2$ and $Z_i \in \{0, 1\}$. Let $Z = \sum_{i=1}^T Z_i$ and $\mu = \sum_{i=1}^T f/2 = E[Z] = 2m \log \Gamma$. Then, for $\Delta = 1/2$, we get

$$\Pr[Z \geq (1 + \Lambda)\mu] \leq \exp\left(-\frac{\Lambda^2}{2 + \Lambda} \cdot 2m \log \Gamma\right) = 2^{-\Omega(m)}.$$

For the second event of staying in the right-side states, Lemma 2 shows that at every round it will succeed producing a block with probability at least $2f$. We view the number of blocks produced as a binomial distribution with success probability $2f$. And, since we assume it begins at the rightmost of `VolatileRightInner`, it has to go leftwards for at most $(m/f) \log \Gamma$ in order to reach `Cold`.

Consider the first $(2m/f) \log \Gamma$ rounds with blocks $\{B_u, \dots, B_v\}$. Since $W_{i+1} = W_i + (r_{i+1} - r_i) - 1/f$, we get $W_v = W_u + r_v - r_u - (u - v)/f$. Assume now $W_u = (m/f) \log 2\Gamma$, if W_v is in `Cold`, $r_v - r_u - (u - v)/f < -(m/f) \log \Gamma$. In expectation, parties will succeed for $(4m \log \Gamma)$ times in $(2m/f) \log \Gamma$ rounds. If they succeed for less than $(3m \log \Gamma)$ times, it cannot satisfy $r_v - r_u - (u - v)/f > -(m/f) \log \Gamma$, thus fail to reach `Cold`, i.e., W_i still falls in `VolatileRightInner`.

By Theorem 19, let Z_i, \dots, Z_T ($T = (2m/f) \log \Gamma$) be independent variables with $E[Z_i] = 2f$ and $Z_i \in \{0, 1\}$. Let $Z = \sum_{i=1}^T Z_i$ and $\mu = \sum_{i=1}^T 2f = E[Z] = 4m \log \Gamma$. Then, for $\Delta = 1/4$, we get

$$\Pr[Z \leq (1 - \Lambda)\mu] \leq \exp\left(-\frac{\Lambda^2}{2 + \Lambda} \cdot 4m \log \Gamma\right) = 2^{-\Omega(m)}.$$

Therefore, the return probability is $1 - 2^{-\Omega(\text{polylog}(\kappa))}$. □

Lemma 3 guarantees “goodness” in a sliding window with fixed length and ideal start state, while Lemma 4 states the desired probability of finding the next ideal start block in such a sliding window. Therefore, we are able to extend our analysis from the first $4(m/f) \log \Gamma$ rounds to the whole execution. Formally:

Theorem 5. All the target recalculation points on a chain in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment are good.

Proof. We show that “goodness” is maintained for all target recalculation points in a sliding window of length $4(m/f) \log \Gamma$, and that it can be iterated from the first round to the whole execution. Note that in our assumption, the first block satisfies $pT_1 n_{r_1} = f$ (i.e., the state is `Cold`), thus, we can establish the first sliding window starting from B_1 . The iteration works as follows: We choose the

last block with state `Cold` in the sliding window, and then establish a new sliding window starting from it. According to Lemma 4, such block always exists. For blocks in each sliding window, Lemma 3 ensures that all the target recalculation points are good (i.e., never go into the state `Hot Left` or `HotRight`). \square

We note that if we consider a respecting environment that allows more time for the same party fluctuation, then the goodness parameter—i.e., the upper bound and lower bound of the target recalculation point—can be closer to f . For example, the good target recalculation parameter can be changed to $(1 + \Gamma^3)f$ and $f/[(2 - \delta)(1 + \Gamma^3)]$, with sliding window length $2\Gamma^3(m/f) \log \Gamma$. In the analysis, this is a trade-off. For simplicity, the presentation in this section chooses the more intuitive values $(2f$ and $f/2)$.

4 Full Security Analysis

So a memoryless target recalculation function, under certain conditions, is able to maintain a steady block generation rate. Can this be used by a protocol to satisfy the desired blockchain and ledger properties, as formulated in [GKL15, GKL17]? That’s what this section demonstrates, by providing the relevant protocol abstraction and following their analytical approach, albeit with different parameters and proofs.

4.1 The Bitcoin Cash Backbone Protocol

The main changes introduced by Bitcoin Cash’s hard fork from Bitcoin were an increase of the block size, replacement of the difficulty adjustment algorithm, and modification of the transaction rules; the protocol structure remained unchanged. For the analysis, we adopt the protocol abstraction presented in [GKL15] (and follow-ups), consisting of the main algorithm (Algorithm 4), which at the beginning of a round receives input (new transactions as well as chains sent by other miners); validates them and compares them (according to their accumulated difficulty) against the miner’s current chain, adopting the one with highest difficulty (it could be the party’s own); and attempts to extend the adopted chain by generating a PoW with the current round’s difficulty value. Before turning to the protocol description, we first review the desired properties the protocol should satisfy.

Blockchain properties. We review the two main properties to be satisfied by a PoW-based blockchain protocol.

- **Common Prefix** Q_{cp} (Parameterized by $k \in \mathbb{N}$): For any two players P_1, P_2 holding chains $\mathcal{C}_1, \mathcal{C}_2$ at rounds r_1, r_2 , with $r_1 \leq r_2$, it holds that $\mathcal{C}_1^{\lceil k} \preceq \mathcal{C}_2$.
- **Chain Quality** Q_{cq} (Parameterized by $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$): For any party P with chain \mathcal{C} in $\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}$, and any segment of that chain of difficulty d such that the timestamp of the first block of the segment is at least ℓ smaller than the timestamp of the last block, the blocks the adversary has contributed in the segment have a total difficulty that is at most $\mu \cdot d$.

Transaction ledger. Similarly to Bitcoin, the main application of the Bitcoin Cash protocol is a *robust transaction ledger*, aimed at maintaining a serialized transaction sequence organized in the form of a blockchain, satisfying the following two properties. Let \mathcal{L} denote such ledger.

- **Consistency:** For any two honest parties P_1, P_2 , reporting $\mathcal{L}_1, \mathcal{L}_2$ at rounds $r_1 \leq r_2$, resp., it holds that \mathcal{L}_1 is a prefix of $\tilde{\mathcal{L}}_2$.

- **Liveness** (Parameterized by $u \in \mathbb{N}$, the “wait time” parameter): If a transaction tx is provided to all honest parties for u consecutive rounds, then it holds that for any player P , tx will be in \mathcal{L} .

Protocol description. As in [GKL15], in our description of the protocol we intentionally avoid specifying the type of values/content that miners try to insert in the chain, the type of chain validation they perform (beyond checking for its structural properties with respect to the hash functions $G(\cdot), H(\cdot)$, and the way they interpret the chain. These checks and operations are handled by the external functions $V(\cdot), I(\cdot)$ and $R(\cdot)$ (the content validation function, the input contribution function and the chain reading function, resp.) which are specified by the application that runs “on top” of the backbone protocol. The Bitcoin Cash protocol in the dynamic setting comprises three algorithms *chain validation*, *chain comparison* and *proof of work*.

Chain validation. The `validate` algorithm performs a validation of the structural properties of a given chain \mathcal{C} . It is given as input the value q , as well as hash functions $H(\cdot), G(\cdot)$. It is parameterized by the content validation predicate $V(\cdot)$ as well as by $D(\cdot)$, the *target calculation function* (see Section 3.1 and Appendix 5). For each block of the chain, the algorithm checks that the proof of work is properly solved (with a target that is suitable as determined by the target calculation function), and that the counter ctr does not exceed q . Furthermore it collects the inputs from all blocks, $\mathbf{x}_{\mathcal{C}}$, and tests them via the predicate $V(\mathbf{x}_{\mathcal{C}})$. Chains that fail these validation procedure are rejected. (Algorithm 1.)

Algorithm 1 The *chain validation predicate*, parameterized by q, T , the hash functions $G(\cdot), H(\cdot)$, and the *content validation predicate* $V(\cdot)$. The input is \mathcal{C} .

```

1: function VALIDATE( $r_{\text{now}}, \mathcal{C}$ )
2:    $valid \leftarrow V(\mathbf{x}_{\mathcal{C}}) \wedge (\mathcal{C} \neq \varepsilon)$ 
3:   if  $valid = \text{true}$  then                                      $\triangleright$  The chain is non-empty and meaningful w.r.t.  $V(\cdot)$ 
4:      $r' \leftarrow r_{\text{now}}$ 
5:      $\langle r, st, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$ 
6:      $st' \leftarrow H(ctr, G(r, st, x))$ 
7:     repeat
8:        $\langle s, x, ctr \rangle \leftarrow \text{head}(\mathcal{C})$ 
9:        $T \leftarrow D(\mathbf{r}_{\mathcal{C}^{\uparrow 1}})$                                       $\triangleright$  Calculate target based on  $\mathcal{C}^{\uparrow 1}$ 
10:      if  $\text{validblock}_q^T(\langle st, x, ctr \rangle) \wedge (H(ctr, G(r, st, x)) = st') \wedge (r < r')$  then
11:         $r' \leftarrow r$                                               $\triangleright$  Retain round timestamp
12:         $st' \leftarrow st$                                             $\triangleright$  Retain hash value
13:         $\mathcal{C} = \mathcal{C}^{\uparrow 1}$                                               $\triangleright$  Remove the head from  $\mathcal{C}$ 
14:      else
15:         $valid \leftarrow \text{False}$ 
16:      end if
17:    until  $(\mathcal{C} = \varepsilon) \vee (valid = \text{False})$ 
18:  end if
19:  return  $valid$ 
20: end function

```

Chain Comparison. The objective of the second algorithm, called `maxvalid`, is to find the “best possible” chain when given a set of chains. The algorithm is straightforward and is parameterized by a $\text{max}(\cdot)$ function that applies some ordering in the space of chains. The most important aspect

is the chains' difficulty in which case $\max(\mathcal{C}_1, \mathcal{C}_2)$ will return the most difficult of the two. In case $\text{diff}(\mathcal{C}_1) = \text{diff}(\mathcal{C}_2)$, some other characteristic can be used to break the tie. In our case, $\max(\cdot, \cdot)$ will always return the first operand to reflect the fact that parties adopt the first chain they obtain from the network. (Algorithm 2.)

Algorithm 2 The function that finds the “best” chain, parameterized by function $\max(\cdot)$. The input is $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$.

```

1: function MAXVALID( $r, \{\mathcal{C}_1, \dots, \mathcal{C}_k\}$ )
2:    $temp \leftarrow \varepsilon$ 
3:   for  $i = 1$  to  $k$  do
4:     if  $\text{validate}(r, \mathcal{C}_i)$  then
5:        $temp \leftarrow \max(\mathcal{C}, temp)$ 
6:     end if
7:   end for
8:   return  $temp$ 
9: end function

```

Algorithm 3 The proof of work function, parameterized by q and hash functions $H(\cdot), G(\cdot)$. The input is (x, \mathcal{C}) .

```

1: function POW( $r, x, \mathcal{C}$ )
2:   if  $\mathcal{C} = \varepsilon$  then                                     ▷ Determine proof of work instance.
3:      $st \leftarrow 0$ 
4:   else
5:      $\langle r', st', x', ctr' \rangle \leftarrow \text{head}(\mathcal{C})$ 
6:      $st \leftarrow H(ctr'G(r', st', x'))$ 
7:   end if
8:    $ctr \leftarrow 1$ 
9:    $B \leftarrow \varepsilon$ 
10:   $T \leftarrow D(\mathbf{r}_{\mathcal{C}})$                                      ▷ Calculate target for next block based on timestamps.
11:   $h \leftarrow G(r, st, x)$ 
12:  while  $(ctr \leq q)$  do
13:    if  $H(ctr, h) < T$  then                                 ▷ Proof of work succeeds and a new block is created.
14:       $B \leftarrow \langle r, st, x, ctr \rangle$ 
15:      break
16:    end if
17:     $ctr \leftarrow ctr + 1$ 
18:  end while
19:   $\mathcal{C} \leftarrow \mathcal{C}B$                                          ▷ Chain is extended.
20:  return  $\mathcal{C}$ 
21: end function

```

Proof of work. The third algorithm, called `pow`, is the proof of work-finding procedure. It takes as input a chain and attempts to extend it via solving a proof of work. This algorithm is parameterized by two hash functions $H(\cdot), G(\cdot)$. Moreover, the algorithm calls the target calculation function $D(\cdot)$ in order to determine the value T that will be used for the proof of work. The procedure, given a chain \mathcal{C} and a value x to be inserted in the chain, hashes these values to obtain h and initializes a counter ctr . Subsequently, it increments ctr and checks to see whether $H(ctr, h) < T$; in case a

suitable ctr is found then the algorithm succeeds in solving the POW and extends chain \mathcal{C} by one block. (Algorithm 3.)

Bitcoin Cash backbone protocol. The core of the Bitcoin Cash backbone protocol with variable difficulty is similar to that in [GKL17], with several important distinctions. First is the procedure to follow when the parties become active. Parties check the **ready** flag they possess, which is false if and only if they have been inactive in the previous round. In case the **ready** flag is false, they diffuse a special message ‘**Join**’ to request the most recent version of the blockchain(s). Similarly, parties that receive the special request message in their RECEIVE() tape broadcast their chains. As before parties, run “indefinitely” (our security analysis will apply when the total running time is polynomial in κ). The input contribution function $I(\cdot)$ and the chain reading function $R(\cdot)$ are applied to the values stored in the chain. Parties check their communication tape RECEIVE() to see whether any necessary update of their local chain is due; then they attempt to extend it via the POW algorithm **pow**. The function $I(\cdot)$ determines the input to be added in the chain given the party’s state st , the current chain \mathcal{C} , the contents of the party’s input tape INPUT() and communication tape RECEIVE(). The input tape contains two types of symbols, READ and (INSERT, $value$); other inputs are ignored. In case the local chain \mathcal{C} is extended the new chain is diffused to the other parties. Finally, in case a READ symbol is present in the communication tape, the protocol applies function $R(\cdot)$ to its current chain and writes the result onto the output tape OUTPUT(). The pseudocode of the backbone protocol is presented in Algorithm 4.

Algorithm 4 The Bitcoin Cash backbone protocol in the dynamic setting at round “**round**” on local state (st, \mathcal{C}) parameterized by the *input contribution function* $I(\cdot)$ and the *chain reading function* $R(\cdot)$. The **ready** flag is **false** if and only if the party was inactive in the previous round.

```

1: if ready = true then
2:   DIFFUSE(‘ready’)
3:    $\tilde{\mathcal{C}} \leftarrow \text{maxvalid}(\mathcal{C} \text{ all chains } \mathcal{C}' \text{ found in RECEIVE}())$ 
4:    $(st, x) \leftarrow I(st, \tilde{\mathcal{C}}, \text{round}, \text{INPUT}(), \text{RECEIVE}())$ 
5:    $\mathcal{C}_{\text{new}} \leftarrow \text{pow}(\text{round}, x, \tilde{\mathcal{C}})$ 
6:   if  $(\mathcal{C} \neq \mathcal{C}_{\text{new}}) \vee (\text{‘Join’} \in \text{RECEIVE}())$  then
7:      $\mathcal{C} \leftarrow \mathcal{C}_{\text{new}}$ 
8:     DIFFUSE( $\mathcal{C}$ )      ▷ chain is diffused when it is updated or when someone wants to join.
9:   end if
10:  if INPUT() contains READ then
11:    write  $R(x_{\mathcal{C}})$  to OUTPUT()
12:    DIFFUSE(RoundComplete)
13:  end if
14: else
15:   ready  $\leftarrow$  true
16:   DIFFUSE(Join, RoundComplete)
17: end if

```

4.2 ASERT: Full Analysis

Table 1 summarizes the parameters that will be used in our analysis, some of which have already been introduced. Note that our security parameter is κ , and $\varphi = \Theta(m) = \text{polylog}(\kappa)$. Moreover, we consider the fluctuation of the total number of parties (cf. Definition 1 and Fact 1).

Table 1: *Summary of parameters (ASERT).*

- | |
|---|
| <ul style="list-style-type: none"> – δ: Advantage of honest parties, $\forall r(t_r/h_r < 1 - \delta)$. – $\gamma, \sigma, \Gamma, \Sigma$: Determine how the number of parties fluctuates across rounds in a period (cf. Definition 1 and Fact 1). – f: Probability that at least one honest party succeeds generating a PoW in a round assuming h_0 parties and target T_0 (the protocol’s initialization parameters). – m: Smoothing factor (cf. Definition 2). – τ: Parameter that regulates the target that the adversary could query the PoW with. – ϵ: Quality of concentration of random variables (cf. Definition 6). – κ: The length of the hash function output. – φ: Related to the properties of the protocol. – L: The total number of rounds in the execution of the protocol. |
|---|

During a round r of an execution E , the honest parties might be split and work on different chains, and thus might query the random oracle on different targets. Denote by T_r^{\min} and T_r^{\max} the minimum and maximum of these targets, respectively.

Next, we extend the definition of “goodness” from Section 3.2 to apply rounds and chains, in addition to recalculation points, and following [GKL17], we define a property called *accuracy*, which we will then show most executions satisfy, and which will help achieve the desired application’s properties.

Definition 4 (Goodness). A target-recalculation point r is *good* if the target T for the next block satisfies $f/2(2 - \delta)\Gamma^3 \leq ph_r T \leq 2\Gamma^3 f$. Round r is *good* if $f/2\gamma(2 - \delta)\Gamma^3 \leq ph_r T_r^{\min}$ and $ph_r T_r^{\max} \leq 2\gamma\Gamma^3 f$. A chain is *good* if all its target-recalculation points are good.

Definition 5 (Accuracy). A block created at round u is *accurate* if it has a timestamp v such that $|u - v| \leq \ell + 2\Delta$. A chain is *accurate* if all its blocks are accurate. A chain is *stale*, if for some $u \geq \ell + 2\Delta$, it does not contain an honest block with timestamp $v \geq u - \ell - 2\Delta$.

For a given round r , we let \mathcal{S}_r denote the set of chains that belong, or could potentially belong to an honest party. Being explicit about this set of chains will help in the formulation of a number of predicates (see below). Specifically, \mathcal{S}_r includes⁴:

- Chain \mathcal{C} that belongs to an honest party;
- chain \mathcal{C} with $\text{diff}(\mathcal{C}) > \text{diff}(\mathcal{C}')$ for some chain \mathcal{C}' of an honest party; and
- chain \mathcal{C} with $\text{diff}(\mathcal{C}) = \text{diff}(\mathcal{C}')$ for some chain \mathcal{C}' of an honest party and $\text{head}(\mathcal{C})$ was computed no later than $\text{head}(\mathcal{C}')$.

Random variables. We are interested in estimating the difficulty acquired by honest parties during a sequence of rounds. For a given round r , the following real-valued random variables are defined in [GKL20]:

- D_r : Sum of the difficulties of all blocks computed by honest parties.
- Y_r : Maximum difficulty among all blocks computed by honest parties.
- Q_r : Equal to Y_r when $D_u = 0$ for all $r < u < r + \Delta$ and 0 otherwise.

⁴Note that these chains should exist and be valid at round r .

A round r such that $D_r > 0$ is called *successful* and one where $Q_r > 0$ *isolated-successful*.

Regarding the adversary, in order to overcome the fact that he can query the oracle for arbitrarily low targets and thus obtain blocks of arbitrarily high difficulty, we would like to upper-bound the difficulty he can acquire during a set J of queries. This is achieved by associating a set of consecutive adversarial queries J with the target of its first query. We denote this target $T(J)$, and say that $T(J)$ is *associated* with J . We then define $A(J)$ and $B(J)$ to be equal to the sum of the difficulties of all blocks computed by the adversary during queries in J for target at least $T(J)/\tau$ and $T(J)$, respectively—i.e., queries in J for targets less than $T(J)/\tau$ (resp. $T(J)$) do not contribute to $A(J)$ (resp. $B(J)$).

For simplicity, we write $h(S) = \sum_{r \in S} h_r$ for a set of rounds S and queries J (similarly, $t(S)$, $D(S)$, $Y(S)$, $Q(S)$, $A(J)$ and $B(J)$).

We also define the random variable \mathcal{E} taking values on our probability space and with a distribution induced by the random coins of all entities (adversary, environment, parties) and the random oracle. Let \mathcal{E}_{r-1} fix the execution just before round r . In particular, a value E_{r-1} of \mathcal{E}_{r-1} determines the adversarial strategy and so determines the targets against which every party will query the oracle at round r and the number of parties h_r and t_r , but it does not determine D_r or Q_r . For an adversarial query j we will use, slightly overloading notation, $E_{j-1}^{(J)}$ to denote the execution just before this query.

The following fact is a consequence of Definition 1 (respecting environments):

Fact 1. Let S be a set of at most Σ consecutive rounds in a $(\langle \gamma, \sigma \rangle, \langle \Gamma, \Sigma \rangle)$ -respecting environment and $U \subseteq S$.

(a) If $U \preceq S$ and $|U| < \sigma$,

$$\frac{h_S}{\Gamma} \leq \frac{h(S)}{|S|} \leq \Gamma \cdot h_S \quad \text{and} \quad \frac{h_U}{\gamma} \leq \frac{h(U)}{|U|} \leq \gamma \cdot h_U,$$

where $h_S \in \{h_r : r \in S\}$ and $h_U \in \{h_r : r \in U\}$.

(b)

$$h(S) \leq \left(1 + \frac{\Gamma|S \setminus U|}{|U|}\right)h(U) \quad \text{and} \quad |S| \sum_{r \in S} (ph_r)^2 \leq \Gamma \left(\sum_{r \in S} ph_r\right)^2.$$

Proof. (a) As proved in [GKL17], the average of several numbers (i.e., $h(S)/|S|$) is bounded by their minimum and maximum, we get the desired inequality.

(b) We learn by (a) that $h(S \setminus U) \leq |S \setminus U| \cdot \Gamma \cdot h(U)/|U|$. Note that $h(S) = h(S \setminus U) + h(U)$, by adding $h(U)$ we get the first inequality. The second inequality is a simple implementation of Theorem 21 with $a_k = ph_r$, $b_k = 1$, $w_k = 1$ and $M = \Gamma' \cdot m$. \square

In order to obtain meaningful concentration of random variables, we have to consider a sufficiently long sequence with a number of rounds at least

$$\ell = \frac{4(2 - \delta)(1 + 3\epsilon)}{\epsilon^2 f [1 - 2\gamma\Gamma^3 f]^{\Delta+1}} \cdot \max\{\Delta, \tau\} \cdot \gamma\Gamma^4 \cdot \varphi. \quad (3)$$

We will assume that ℓ is appropriately small compared to the length m of a sliding interval/window. Specifically,

$$2\ell + 6\Delta \leq \frac{\epsilon m}{2\gamma\Gamma^3 f}. \quad (C1)$$

In addition, we would like the advantage δ of the honest parties over adversarial parties to be large enough to absorb error factors. Thus, we require the following inequalities:

$$[1 - 2\gamma\Gamma^3 f]^\Delta \geq 1 - \epsilon \quad \text{and} \quad \epsilon \leq \delta/8 \leq 1/8. \quad (C2)$$

Next, we show a chain-growth lemma referring to accumulated difficulty (cf. [GKL20]), as opposed to number of blocks in the original formulations [GKL15, KP15].

Lemma 6 (Chain Growth). *Suppose that at round u of an execution E an honest party diffuses a chain of difficulty d . Then, by round v , every honest party has received a chain of difficulty at least $d + Q(S)$, where $S = \{r : u + \Delta \leq r \leq v - \Delta\}$.*

Proof. If two blocks are obtained at rounds which are at distance at least Δ , then we are certain that the later block increased the accumulated difficulty. To be precise, assume $S^* \subseteq S$ is such that, for all $i, j \in S^*$, $|i - j| \geq \Delta$ and $Y_i > 0$. We argue that, by round v , every honest party has a chain of difficulty at least

$$d + Y(S^*) \geq d + Q(S).$$

Observe first that every honest party will receive the chain of difficulty d by round $u + \Delta$ and so the first block obtained in S^* extends a chain of weight at least d . Next, note that if a block obtained in S^* is the head of a chain of weight at least d' , then the next block in S^* extends a chain of weight at least d' . \square

Typical executions. The notion of *typical executions* was introduced in the analysis framework we are following [GKL15] in order to capture situations where an execution E 's progress does not deviate too much from its expected (desired) progress. Since executions consist of rounds, and within rounds parties perform Bernoulli trials, we can calculate the expected progress when given the corresponding probabilities. On this basis, if the difference between the real execution and its expectation is reasonable, E is declared “typical.” Note that besides expectation, the variance should also be taken into consideration. We will later show (applying Theorem 20—martingale inequality) that either the variance is too high with respect to a set of rounds, or the parties have made progress during these rounds as expected.

In addition to the behavior of the random variables described above, bad events may occur related to the underlying hash function $H(\cdot)$, which is modeled as a random oracle and used to obtain PoWs. The bad events are *insertion* (of a block in between two consecutive blocks), *copy* (same block exists in two different position of the blockchain), and *prediction* (a block extends one with an earlier creation time). Refer to [GKL20] for a precise definition. A typical execution will rule out these bad events as well.

We are now ready to specify what is a typical execution in our setting (compare with [GKL20]'s).

Definition 6 (Typical execution). An execution E is *typical* if the following hold:

- (a) For any set S of at least ℓ consecutive good rounds,

$$(1 - \epsilon)[1 - 2\gamma\Gamma^3 f]^\Delta ph(S) < Q(S) \leq D(S) < (1 + \epsilon)ph(S).$$

- (b) For any set J indexing a set of consecutive adversarial queries and $\alpha(J) = 2(\frac{1}{\epsilon} + \frac{1}{3})\varphi/T(J)$,

$$A(J) < p|J| + \max\{\epsilon p|J|, \tau\alpha(J)\} \text{ and } B(J) < p|J| + \max\{\epsilon p|J|, \alpha(J)\}.$$

- (c) No insertions, no copies, and no predictions occurred in E .

The next proposition is a simple application of Definition 6 and the honest-majority assumption.

Proposition 7. *Let E be a typical execution in a $(\langle\gamma, \sigma\rangle, \langle\Gamma, \Sigma\rangle)$ -respecting environment. Let $S = \{r \mid (u \leq r \leq v) \wedge (v - u \geq \ell)\}$ be a set of consecutive good rounds and J the set of adversarial queries in $U = \{r \mid u - \Delta \leq r \leq v + \Delta\}$. Then the following inequalities hold:*

- (a) $(1 + \epsilon)p|J| \leq Q(S) \leq D(U) < (1 + 5\epsilon)Q(S)$.
(b) If w is a good round such that $|w - r| \leq \Sigma$ for any $r \in S$, then $Q(S) > (1 + \epsilon)[1 - 2\gamma\Gamma^3 f]^\Delta |S| ph_w / \Gamma$. If, in addition, $T(J) \geq T_w^{\min}$, then $A(J) < (1 - \delta + 3\epsilon)Q(S)$.

Proof. (a) The middle inequality directly follows the definition of the random variables Q and D . For the other two inequalities, since $|J| \leq (1 - \delta)h(U)$ we only have to compare the value of $h(S)$ and $h(U)$. Note that Fact 1(b) implies that $h(U \setminus S) \leq \Gamma \cdot |U \setminus S| \cdot \frac{h(S)}{|S|} \leq \Gamma \cdot \frac{h(S)}{\ell} \cdot 2\Delta$. Thus,

$$h(U) \leq (1 + \frac{2\Gamma\Delta}{\ell})h(S) < (1 + \frac{\epsilon^2}{2})h(S).$$

(b) The first inequality is a direct application of Fact 1(a). For the second one, round w is good implies $ph_w T(J) \geq ph_w T_w^{\min} \geq f/2\gamma(2 - \delta)\Gamma^3$. Condition (C2) implies that

$$\epsilon(1 - 2\epsilon)ph(S) \geq \epsilon(1 - 2\epsilon)|S| \cdot \frac{ph_w T_w^{\min}}{\Gamma T(J)} \geq \frac{\epsilon(1 - 2\epsilon)f\ell}{2\gamma(2 - \delta)\Gamma^4 T(J)} > \tau\alpha(J).$$

As showed in (a), $p|J| \leq (1 - \delta + \epsilon^2/2)ph(S)$. For either $\epsilon p|J|$ or $\tau\alpha(J)$ in Definition 6(b), we obtain $A(J) < (1 - \delta + 3\epsilon)Q(S)$. \square

We are now able to show that almost all Bitcoin Cash backbone protocol executions polynomially bounded (in κ) are typical (the proof is presented in Appendix C). Formally:

Theorem 8. *Assuming the Bitcoin Cash backbone protocol runs for L rounds, the event “ E is not typical” is bounded by $\text{poly}(L) \cdot e^{-\Omega(\text{polylog}(\kappa))}$.*

Accuracy and goodness. Next, we consider accuracy and goodness over the space of typical executions in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment, as well as implications between the two. We assume that all the requirements for the initialization parameters h_0 and T_0 are satisfied.

Lemma 9. *Let E be a typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment. If E_{r-1} is good, then there are no stale chains in \mathcal{S}_r .*

Proof. For the sake of a contradiction, consider a chain $\mathcal{C} \in \mathcal{S}_r$ which has not been extended by the honest party for at least $\ell + 2\Delta$ rounds. Without loss of generality, let r denote the least round with this property. Let B be the last block of \mathcal{C} computed by the honest party (possibly the genesis block) and let w be its timestamp. Set $S = \{u : w + \Delta \leq u \leq r - \Delta\}$ and $U = \{u : w \leq u \leq r\}$. Therefore, to reach a contradiction it suffices to show that the adversary’s accumulated difficulty $d < Q(S)$.

Let J denote the queries in U starting from the first adversarial query attempting to extend B . We learn from Condition (C1) that targets the adversary can query during U is bounded by $T(J)/\tau$. Thus, $d < A(J)$. If $A(J) < (1 + \epsilon)p|J|$, then $A(J) < Q(S)$ is obtained by Proposition 7(a). Otherwise, $A(J) < (\frac{1}{\epsilon} + 1)\tau\alpha(J) = 2(\frac{1}{\epsilon} + 1)(\frac{1}{\epsilon} + \frac{\epsilon}{3})\tau\varphi/T(J)$. However, by considering only the first ℓ rounds in S , $h(S) \geq h_u \ell / \gamma$. We have

$$Q(S) > (1 - \epsilon)[1 - 2\gamma\Gamma^3 f]^\Delta \cdot \frac{ph_u \ell T(J)}{\gamma T(J)} > \frac{(1 - \epsilon)[1 - 2\gamma\Gamma^3 f]^\Delta f \ell}{2\gamma(2 - \delta)\Gamma^4 T(J)} \geq \frac{2(1 - \epsilon)(1 + 3\epsilon)\tau\varphi}{\epsilon^2 T(J)} \geq A(J).$$

In either situation, we obtain the desired inequality $d < Q(S)$. \square

Corollary 10. *Let E be a typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment. If E_{r-1} is good, then all chains in \mathcal{S}_r are accurate.*

Proof. Suppose—towards a contradiction—that, for some $w \leq r$, $\mathcal{C} \in \mathcal{S}_r$ contains a block which is not accurate and let $u \leq w$ be the timestamp of this block and v its creation time. If $u - v > \ell + 2\Delta$, then every honest party would consider \mathcal{C} to be invalid during rounds $v, v + 1, \dots, u$. If $v - u > \ell + 2\Delta$, then in order for \mathcal{C} to be valid it should not contain any honest block with timestamp in $u, u + 1, \dots, v$. (Note that we are using Definition 6(c) here as a block could be inserted later.) In either case, $\mathcal{C} \in \mathcal{S}_r$ but is stale, contradicting Lemma 9. \square

Theorem 11. *A typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment is accurate and good.*

Proof. Accuracy directly follows Corollary 10. Regarding goodness, Theorem 5 claims that all the target recalculation points are good. For those rounds which are not target recalculation points, it follows from Lemma 9 that the maximum interval between two blocks is $\ell + 2\Delta$, thus, by Fact 1(a) $h_u/\gamma \leq h_r \leq \gamma h_u$ holds. Combining it with $f/2(2 - \delta)\Gamma^3 \leq ph_u T \leq 2\Gamma^3 f$, we obtain the desired inequality in Definition 4. \square

Blockchain and ledger properties. We now show that the Bitcoin Cash backbone protocol satisfies the two properties common prefix and chain quality (Section 4.1) for a suitable respecting environment. First, a preliminary lemma:

Lemma 12. *For any round r of a typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment and any two chains \mathcal{C} and \mathcal{C}' in \mathcal{S}_r , the timestamp of $\text{head}(\mathcal{C} \cap \mathcal{C}')$ is at least $r - 2\ell - 4\Delta$.*

Proof. Let v be the timestamp of $\text{head}(\mathcal{C} \cap \mathcal{C}')$ and $u \leq v$ the greatest timestamp among those blocks on $\mathcal{C} \cap \mathcal{C}'$ that was computed by an honest party. Let $U = \{i : u < i \leq r\}$, $S = \{i : u + \Delta \leq i \leq r - \Delta\}$, and let J denote the adversarial queries that correspond to the rounds in U . We claim that if $r - v \geq \ell + 2\Delta$, then

$$2Q(S) \leq D(U) + A(J).$$

This contradicts Proposition 7 for $\delta \geq 8\epsilon$, which implies $D(U) < (1 + 5\epsilon)Q(S)$ and $A(J) < (1 - \delta + 3\epsilon)Q(S)$.

Towards proving the claim, associate with each $r \in S$ such that $Q_r > 0$ an arbitrary honest block that is computed at round r for difficulty Q_r . Let \mathcal{B} be the set of these blocks and note that their difficulties sum to $Q(S)$. We argue the existence of a set of blocks \mathcal{B}' computed in U such that $\mathcal{B} \cap \mathcal{B}' = \emptyset$ and $\{d \in \mathcal{B} : B \in \mathcal{B}\} \subseteq \{d \in \mathcal{B} : B \in \mathcal{B}'\}$. This suffices, because each block in \mathcal{B}' contributes either to $D(U) - Q(S)$ or to $A(J)$ and so $Q(S) \leq D(U) - Q(S) + A(J)$.

Consider, then, a block $B \in \mathcal{B}$ extending a chain \mathcal{C}^* and let $d = \text{diff}(\mathcal{C}^*B)$. If $d \leq \text{diff}(\mathcal{C} \cap \mathcal{C}')$ (note that $u < v$ in this case and $\text{head}(\mathcal{C} \cap \mathcal{C}')$ is adversarial), let B' be the block of $\mathcal{C} \cap \mathcal{C}'$ containing d . Such a block clearly exists and has a timestamp greater than u . Furthermore, $B' \notin \mathcal{B}$, since B' was computed by the adversary. If $d > \text{diff}(\mathcal{C} \cap \mathcal{C}')$, note that there is a unique $B \in \mathcal{B}$ such that $d \in B$ (recall the argument in Chain Growth Lemma). Since B cannot simultaneously be on \mathcal{C} and \mathcal{C}' , there is a $B' \notin \mathcal{B}$ either on \mathcal{C} or on \mathcal{C}' that contains d . \square

Theorem 13 (Common-Prefix). *For a typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment, the common-prefix property holds for parameter ϵm .*

Proof. Suppose—towards a contradiction—common prefix fails for two chains \mathcal{C}_1 and \mathcal{C}_2 at rounds $r_1 \leq r_2$. It is easy to see that there exist a round $r \leq r_2$ and two chains \mathcal{C} and \mathcal{C}' such that each had at least k blocks after $\text{head}(\mathcal{C} \cap \mathcal{C}')$. In view of Lemma 12, it suffices to show that these blocks were computed within at least $\ell + 2\Delta$ rounds. Suppose the honest parties query during a set of rounds S of size $\ell + 2\Delta$, the accumulated difficulty is $D(S)$, and the number of blocks produced in $|S|$ is at most $T^{\max} \cdot D(S)$ where T^{\max} denote the maximum target among blocks in $|S|$. For the target recalculation round r^* associated with target T^{\max} , we have

$$T^{\max} \cdot D(S) < (1 + \epsilon)pT^{\max}h(S) \leq (1 + \epsilon)p\gamma T^{\max}h_{r^*}|S| < (1 + \epsilon)\epsilon m/2.$$

For the last inequality, note that by Condition (C1) $|S| < \epsilon m/(4\gamma\Gamma^3 f)$ and by Theorem 11 $pT^{\max}h_{r^*} \leq 2\Gamma^3 f$. By proposition 7 the adversary contributed to less than $(1 + \epsilon)\epsilon m/2(1 + \delta)$ blocks, for a total of less than ϵm . \square

Theorem 14 (Chain-Quality). *For a typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment, the chain-quality property holds with parameters $\ell + 2\Delta$ and $\mu = \delta - 3\epsilon$.*

Proof. Let us denote by B_i the i -th block of \mathcal{C} so that $\mathcal{C} = B_1 \dots B_{\text{len}(\mathcal{C})}$ and consider K consecutive blocks B_u, \dots, B_v . Define K' as the least number of consecutive blocks $B_{u'} \dots B_{v'}$ that include the K given ones (i.e., $u' \leq u$ and $v \leq v'$) and have the properties (1) that the block $B_{u'}$ was computed by an honest party or is B_1 in case such block does not exist, and (2) that there exists a round r' that $B_1 \dots B_{v'} \in \mathcal{S}_{r'}$. Denote by d' the total difficulty of these K' blocks. Define $U = \{r, \dots, r'\}$, $S = \{r + \Delta, \dots, r' - \Delta\}$, and J the adversarial queries in U starting with the first to obtain one of the K' blocks. Let x denote the total difficulty of all the blocks from honest parties that are included in the K blocks and—towards a contradiction—assume $x < \mu d'$. In a typical execution, all the K' blocks $B_j : u' \leq j \leq v'$ have been computed in U . But then we have the following contradiction to Proposition 7(b).

$$(1 - \delta + 3\epsilon)Q(S) > A(J) \geq d' - x \geq (1 - \mu)d' \geq (1 - \mu)Q(S) = (1 - \delta + 3\epsilon)Q(S).$$

The inequalities follow from the definitions of x and d' and Chain-Growth Lemma. Note that if $U > 4(m/f) \log \Gamma$, we may use Lemma 9 to partition U appropriately and apply Proposition 7(b) to each part. Its validity come from that a block computed by an honest party provides both properties (1) and (2) required for K' . \square

We conclude by showing that a typical execution of the Bitcoin Cash backbone protocol in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment, satisfies the two properties of a robust transaction ledger presented in Section 2. They are the direct consequence of the blockchain properties shown above, following the approach in [GKL17, GKL20].

Theorem 15 (Consistency). *For a typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment, Consistency is satisfied by setting the settled transactions to be those reported more than ϵm blocks deep.*

Theorem 16 (Liveness). *For a typical execution in a $(\langle \gamma, \ell + 2\Delta \rangle, \langle \Gamma, 4(m/f) \log \Gamma \rangle)$ -respecting environment, Liveness is satisfied for depth ϵm with wait-time $(4\Gamma^4 + 1)\epsilon m/f$.*

Proof. We claim that the chain \mathcal{C} of any honest party has at least ϵm blocks that were computed in the last $2(2 - \delta)\epsilon\Gamma^4 m/(1 - 2\epsilon)f + 2\Delta$ rounds. Denote by T^{\min} the lowest target among these

blocks and r^* a round associated with this target. If S is its subset without the first and last Δ rounds, by Chain-Growth Lemma, the length of the segment is at least

$$T^{\min} \cdot Q(S) > (1 - \epsilon)[1 - 2\gamma\Gamma^3 f]^\Delta p T^{\min} h(S) \geq \frac{(1 - 2\epsilon)f|S|}{2(2 - \delta)\Gamma^4} \geq \epsilon m.$$

Furthermore, if a transaction tx is included in any block computed by an honest party for the first $\ell + 2\Delta$ rounds, by Lemma 9, the chain \mathcal{C} of any honest party contains tx in a block B . Thus, the honest parties will have

$$\ell + 6\Delta + \frac{2(2 - \delta)\epsilon\Gamma^4 m}{(1 - 2\epsilon)f} \leq \frac{\epsilon m}{2\gamma\Gamma^3 f} + \frac{2(2 - \delta)\epsilon\Gamma^4 m}{(1 - 2\epsilon)f} \leq (4\Gamma^4 + 1) \cdot \frac{\epsilon m}{f}$$

as the total wait-time. □

5 The SMA Target Recalculation Function

For completeness, here we present a brief description and analysis of Bitcoin Cash's previous target recalculation function. Even though this function has now been deprecated, it provides some insights and elements of comparison against ASERT based on actual party fluctuation. The full analysis of SMA is presented in Appendix D.1.

Recall that the target calculation function $D(\cdot)$ aims at maintaining the block production rate constant. The probability $f(T, n)$ with which n parties produce a new block with target T is approximated by

$$f(T, n) \approx \frac{qTn}{2^\kappa}.$$

As in the case of Bitcoin, to achieve the above goal Bitcoin Cash tries to keep $qTn/2^\kappa$ close to f . To that end, the SMA function watches an *epoch* of m previous blocks, and based on their difficulty as well as on how fast these blocks were generated, it computes the next target. More specifically, say the last m blocks of a chain \mathcal{C} with targets $\{T_i\}_{i \in [m]}$ were produced in Λ rounds. For every block in the epoch and n participants, it holds that $f(T_i, n) \approx qT_i n / 2^\kappa$. For m consecutive blocks, the average block generating rate $f^* = qn \sum_{i \in [m]} T_i / (m \cdot 2^\kappa)$, and the entire generating time $\Lambda = m / f^* = m^2 \cdot 2^\kappa / (qn \sum_{i \in [m]} T_i)$.

Consider the case where a number of parties

$$n(T_1, \dots, T_m, \Lambda) = \frac{m^2 \cdot 2^\kappa}{q\Lambda \sum_{i \in [m]} T_i}$$

attempt to produce m blocks of target $\{T_i\}_{i \in [m]}$ in Λ rounds (in expectation). Such number of players can be estimated as $n(T_1, \dots, T_m, \Lambda) = m^2 \cdot 2^\kappa / (q\Lambda \sum_{i \in [m]} T_i)$; then the next target T' is set so that $n(T_1, \dots, T_m, \Lambda)$ players would need $1/f$ rounds in expectation to produce the next block of target T' . Therefore, it makes sense to set

$$T' = \frac{\Lambda}{m^2/f} \cdot \sum_{i \in [m]} T_i$$

because if the number of players is indeed $n(T_1, \dots, T_m, \Lambda)$ and remains unchanged, it will take them $1/f$ rounds in expectation to produce next block. If the estimate of the initial number of parties is n_0 , we will assume T_0 is appropriately set so that $f \approx qT_0 n_0 / 2^\kappa$ and then

$$T' = \frac{n_0}{n(T_1, \dots, T_m, \Lambda)} \cdot T_0.$$

Based on the above, we can now give a formal definition of the SMA function. In the definition, parameter τ serves as the “dampening filter” to deal with the case $n_0 T_0 / n(T^{\text{avg}}, \Lambda) \notin [T/\tau, \tau T]$. This mechanism is necessary given an efficient attack presented by Bahack [Bah13] for Bitcoin, which also applies to Bitcoin Cash.

Definition 7 (SMA). For fixed constants $\kappa, \tau, m, n_0, T_0$, the target calculation function $D_{\text{SMA}} : \mathbb{Z}^* \rightarrow \mathbb{R}$ is defined as

$$D_{\text{SMA}}(\varepsilon) = T_0 \text{ and } D_{\text{SMA}}(r_1, \dots, r_v) = \begin{cases} \frac{1}{\tau} \cdot T^{\text{avg}}, & \text{if } \frac{n_0 \cdot T_0}{n(T^{\text{avg}}, \Lambda)} < \frac{1}{\tau} \cdot T^{\text{avg}} \\ \tau \cdot T^{\text{avg}}, & \text{if } \frac{n_0 \cdot T_0}{n(T^{\text{avg}}, \Lambda)} > \tau \cdot T^{\text{avg}} \\ \frac{n_0}{(T^{\text{avg}}, \Lambda)} \cdot T_0, & \text{otherwise} \end{cases}$$

where $n(T^{\text{avg}}, \Lambda) = \frac{m \cdot 2^\kappa}{q \Lambda T^{\text{avg}}}$, with $\Lambda = r_{v-1} - r_{v-m}$ and $T^{\text{avg}} = \frac{1}{m} \cdot \sum_{i=1}^m D(r_1, \dots, r_{v-i})$.

Remark 1. At the start of the protocol execution—the first m blocks – parties cannot acquire enough previous blocks for target recalculation. While several alternate approaches exist (e.g., maintain a static target or switch to other target recalculation function), we will assume a “safe start”, i.e., there exist blocks with an ideal target value prior to the first round.

5.1 SMA: Analysis Overview

The difference between SMA and Bitcoin’s target recalculation is that the target variation (targets that the adversary can query during a sliding window) is no longer bounded by the dampening filter (τ). This is because in SMA the target is recalculated for *every block*; τ only sets restriction on a single calculation.

To highlight some aspects of the proofs of security of the Bitcoin Cash backbone protocol using SMA, we show that while the targets in an epoch keep varying, we can still always find an appropriate block such that the difficulty accumulated during the epoch (when associated with this block) is well bounded. I.e., even the adversary and an honest party join force, they cannot accelerate this process too much; and even when honest parties work independently, they can still efficiently produce enough blocks. The details can be found in Lemma 27.

Moreover, we would like to show that a good relation between the accumulated difficulty in an epoch and its duration always holds, and therefore the target for the next block is good and the block generating rate is close to f . Since the SMA function employs the average targets in the epoch, the previous argument (which only takes into account single target value) collapses. In order to absorb the new errors, We propose new conditions such as the following:

$$\frac{4\lambda\Gamma}{(\lambda\Gamma + 1)^2} \cdot [1 - (1 + \delta)\gamma\Gamma f]^\Delta > 1 - \epsilon \text{ and } \epsilon \leq \delta/8 \leq 1/8 \quad (\text{C4})$$

where $4\lambda\Gamma/(\lambda\Gamma + 1)^2$ (cf. Theorem 22) helps to “revert” the discrepancy between the accumulated difficulty and the next target. (Details in Lemma 28.)

We conclude that the Bitcoin Cash backbone protocol with the SMA function satisfies Consistency and Liveness using the following parameters:

Theorem 17 (Consistency). *For a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, Consistency is satisfied by setting the settled transactions to be those reported more than $\epsilon\gamma^2 m/2\Gamma$ blocks deep.*

Theorem 18 (Liveness). *For a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, Liveness is satisfied for depth $\epsilon\gamma^2 m/2\Gamma$ with wait-time $(\gamma^2 + 1)\epsilon m/f$.*

6 Comparison with the Bitcoin Cash Network

Our analysis shows that for a sufficiently long execution in a somewhat idealized network environment and without severe external disturbances, Bitcoin Cash would achieve the desired security properties (with overwhelming probability). Comparison of our analysis with the existing Bitcoin Cash network data reveals that the ASERT function performs better than the SMA function in an environment where party fluctuation is large. Further, both SMA and ASERT suffer from undersized parameters. In this section we expand on the above comparisons.

6.1 The Bitcoin Cash Network

Party fluctuation. To perform our comparison, we need to determine the fluctuation (ratio) of the number of parties in the actual Bitcoin Cash network. We can extract it from the public statistical data—the *hashrate*, which is the number of hash queries that all miners in the network perform per second. In our model, the queries that every party can make in one round is bounded. Thus, for the purpose of our comparison we assume the hashrate to be identical to the number of parties, and its fluctuation ratio also reflects the the number of parties’. To be more precise, the available hashrate does not reveal the entire Bitcoin Cash network, but instead the hashing power invested in some well-known mining pools. Nonetheless, since most of the computational power is concentrated in these mining pools, this hashrate can very closely approximate the entire network’s.

In addition, instead of the the exact fluctuation ratio with respect to a short period of time (e.g., 10 minutes), we adopt the ratio based on the *daily average hashrate*, which shows the average queries per second in one day; this measure would correspond to parameter Γ in our analysis. Since our analysis should be applied to a relatively long execution, this measure replacement seems reasonable.⁵

Figure 2: *The daily average hashrate of Bitcoin Cash from July 18 to August 16, 2020.*

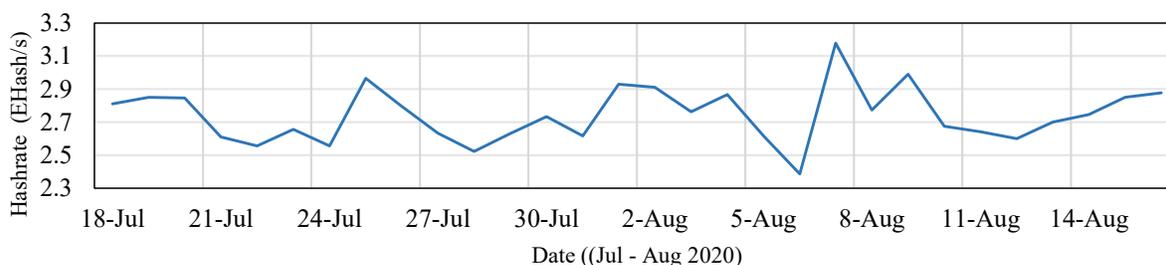


Figure 2 shows the *daily average hashrate* in one month period (Jul 18 2020 – Aug 17 2020)⁶, which we adopt as the representative case of the Bitcoin Cash network execution under the SMA

⁵We note that if we consider the fluctuation rate over 10-minute intervals, the overall party fluctuation will become extremely pronounced (parameter Γ will exceed 4), making our analysis inapplicable.

⁶Source: <https://bitinfocharts.com/comparison/hashrate-bch.html>

function. The maximum value in this period is 3.1783EHash/s (on Aug 07 2020) and the minimum is 2.3865EHash/s (on Aug 06 2020), so it suffices to set $\Gamma = 1.398$ as the party fluctuation ratio. Regarding the ratio during a small period of time, note that as we are discussing the average hashrate, $\gamma = 1.057$, which satisfies $\gamma^{[\Sigma/\sigma]} > \Gamma$, would be a suitable value.

We note that the assumed behavior may not hold when some type of external disturbance occurs, such as the halving of the block reward or sharp gains and declines in BCH’s monetary value, which might cause a large number of miners abruptly joining or dropping out of a mining pool in a short period of time. As it turns out, these events do not happen very frequently (e.g., block reward halves about every 4 years), so we are comfortable extrapolating the execution of the Bitcoin Cash network from what Figure 2 depicts.

Figure 3: *The daily average hashrate of Bitcoin Cash from Dec 1 to Dec 30, 2020.*

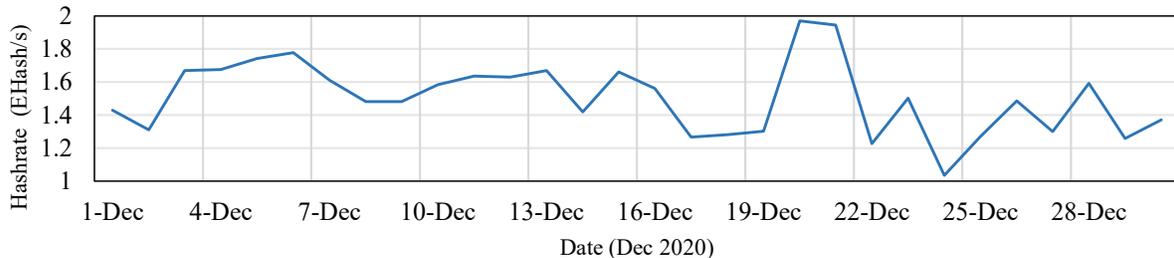


Figure 3 shows the daily average hashrate in one month period (Dec 1 2020 – Dec 30 2020), after the Nov. 2020 network update when the ASERT function was adopted. The maximum value in this period is 1.9451EHash/s (on Dec 21 2020) and the minimum is 1.0349EHash/s (on Dec 24 2020). Therefore, Γ should be to 1.88. This fluctuation rate is relatively high, and we speculate was caused by two reasons: (1) the network could not return to a quiet state soon after the update happens, and (2) the price of Bitcoin Cash experienced substantial ups and downs during this period. We thus set the value of $\gamma = 1.099$.

Network delay. Another important aspect to validate our analysis in a bounded-delay network environment is the actual message delay in the network. It is shown in [DW13] that the delay in the Bitcoin network mainly stems from its multi-hop broadcast (“diffuse” in our model terminology) and block propagation mechanism (Bitcoin Cash employs a similar mechanism to Bitcoin), which we now expand on. As a mining node would typically maintain a limited connections with other nodes, when the node wants to broadcast one block after receiving and verifying it, the node will first send an *inv* message containing the block hash to all its neighbors. If the neighbors do not have the block locally, they will issue a *getdata* message to the sender of *inv*. Then the actual block transfer begins. Thus, at each hop during the broadcast the message incurs a propagation delay consisting of the transmission time and the local verification of the block. Decker and Wattenhofer [DW13] evaluated the distribution of the block propagation time since the first block announcement, which shows a median time of 6.5 seconds as well as a mean time of 12.6 seconds. For more recent data, Bitcoin Monitoring⁷ carried out by the German Federal Ministry of Education and Research shows that 90% of the block propagation time is below 6 seconds. Thus, assuming a round duration of 6 seconds, it is reasonable to let the delay bound (Δ) in our analysis approximately equal to 1 round.

⁷Source: <https://dsn.kastel.kit.edu/bitcoin/>

Target variation. Our formulation bounds the target variation by a fixed parameter λ and the parties’ fluctuation ratio. This is a reasonable idea, since the number of parties, accumulated difficulty and number of blocks (chain growth) are correlated. In the real network, however, the hash power invested in minor blockchains varies wildly, and as such it is unlikely that the number of parties would stay high for a long period of time, and thus target values would seldom vary more pronouncedly than the number of parties. Compared with other parameters, the criteria for selecting λ does not seem as strict. As we can cover more executions if we set a larger value for λ , we choose $\lambda = 1.201$ when applying our analysis. As a result, when compared with previous work [GKL17, GKL20], our analysis can tolerate situations where targets vary more wildly (i.e., they exceed the $\lambda\Gamma$ bound).

6.2 Conclusions

ASERT function. Based on the considerations above, for the purpose of our comparison, we parameterize the real-world Bitcoin Cash network with network bounded-delay $\Delta = 1$ (round), honest advantage $\delta = 0.99$, quality of concentration $\epsilon = 0.123$, long term party fluctuation ratio $\Gamma = 1.88$, short term party fluctuation ratio $\gamma = 1.099$ and ideal block generating rate $f = 0.01$.

On one hand, the resulting probabilities with respect to the goodness parameters are very tight, when considered in the real network parameters (i.e., $m = 288$ and $\Gamma = 1.88$). To be precise, in a sliding window, the probability of the block generation rate exceeding the goodness bound is below 10^{-12} , and the probability of not returning to Cold is less than 10^{-9} . Therefore, it is safe to conclude that the ASERT function achieves a relatively stable block generation rate.

On the other hand, we cannot directly plug in in these parameter values to satisfy Condition (C2), since the party fluctuation is too pronounced, thus making it hard for the concentration parameter ϵ to absorb the errors. Condition (C2), however, can be satisfied in the following two scenarios: (1) party fluctuation ratio in a relatively quiet environment (e.g., $\Gamma = 1.398$ as shown in Fig 2). Then all the requirements are satisfied and so does our analysis. (2) Balancing the block generation rate bound and the sliding window length we consider (cf. Section 3.2), then all the requirements are satisfied if the upper bound for goodness is $(1 + \Gamma^3)f$.

Finally, we observe that the choice of the smoothing factor (m) is too low. While the current value $m = 288$ can satisfy the basic requirements in Condition (C1), it is better to let $m = 432$ to increase the value of ℓ , thus making it closer to the real execution. However, we note that such value of m still fails to give a tight typical execution probability. This is because in order to satisfy the (desired) blockchain properties, the length of a sliding window should be much larger (of the order of years!) so that the martingale probability in Theorem 8 can be negligible.

SMA function. We consider the same network delay, honest advantage and quality of concentration as those used for the ASERT evaluation. Afterwards, we parameterize the SMA function with long-term party fluctuation ratio $\Gamma = 1.398$, short-term party fluctuation ratio $\gamma = 1.057$ and target fluctuation parameter $\lambda = 1.201$. We obtain that Condition (C4) is satisfied under these parameters.

Regarding the protocol parameters m and τ in use, $m = 144$ meets the lowest criteria to support our analysis, $\tau = 2$ may fail in some situations, which can be avoided by these two parameters. More specifically, with regard to the epoch length m , from Condition (C3) we get an upper bound for ℓ of about 640 seconds, which would not be desirable as it exceeds the expected block production interval (600s). We would like that the actual value of ℓ amounts at least to a few blocks’ total expected generation time. Thus, setting $m = 432$ (almost 3 days) would yield an ideal epoch length. Regarding the dampening filter τ , it follows from our analysis that it should hold that

$\tau \geq 2(1 + \delta)\Gamma^2 \approx 7.8$. Thus, $\tau = 8$ would be a suitable value.

ASERT vis-à-vis SMA The above evaluation of the SMA function reveals two shortcomings. Under some circumstance, letting $\lambda = 1.201$ may fail to bound the target fluctuation during a sliding window. Moreover, we can see that parameters plugged into Condition (C4) cannot be increased, i.e., the protocol in our analysis cannot be secure under a respecting environment with party fluctuation ratio $\Gamma > 1.4$; if we consider a larger value for λ , then the party fluctuation it can tolerate is even smaller.

Compared with the ASERT function (notably, ASERT can work well when $\Gamma = 1.88$), SMA performs considerably worse when party fluctuation rate is relatively high. Thus, we conclude that ASERT is a better choice as a target recalculation function for blockchains with lesser hashing power.

References

- [Bah13] Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). Cryptology ePrint Archive, Report 2013/868, 2013. <https://eprint.iacr.org/2013/868>.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS '93*, page 62–73, New York, NY, USA, 1993. Association for Computing Machinery.
- [Can00a] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.
- [Can00b] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. <https://eprint.iacr.org/2000/067>.
- [Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS '01*, page 136, USA, 2001. IEEE Computer Society.
- [DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 139–147, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.
- [DW13] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, 2013.
- [GKL14] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. Cryptology ePrint Archive, Report 2014/765, 2014. <https://eprint.iacr.org/2014/765>.
- [GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, Berlin, Heidelberg, 4 2015. Springer Berlin Heidelberg.

- [GKL17] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 291–323, Cham, 2017. Springer International Publishing.
- [GKL20] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. Full analysis of nakamoto consensus in bounded-delay networks. Cryptology ePrint Archive, Report 2020/277, 2020. <https://eprint.iacr.org/2020/277>.
- [HT94] Vassos Hadzilacos and Sam Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical report, 1994.
- [KP15] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. <https://eprint.iacr.org/2015/1019>.
- [McD98] Colin McDiarmid. Probabilistic methods for algorithmic discrete mathematics, chapter concentration, pages 195–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.
- [MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
- [Nak09a] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. <http://www.bitcoin.org/bitcoin.pdf>.
- [Nak09b] Satoshi Nakamoto. Bitcoin open source implementation of p2p currency, February 2009. <http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source>.
- [PSS17] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 643–673, Cham, 2017. Springer International Publishing.
- [WAT55] G. S. WATSON. Serial correlation in regression analysis i. *Biometrika*, 42(3-4):327–341, 1955.
- [WISK20] Sam M. Werner, Dragos I. Ilie, Iain Stewart, and William J. Knottenbelt. Unstable throughput: When the difficulty algorithm breaks, 2020.

A Mathematical Facts

Theorem 19 (Chernoff bound). *Let X_1, \dots, X_T be independent random variables with $\mathbb{E}[X_i] = p_i$ and $X_i \in [0, 1]$. Let $X = \sum_{i=1}^T X_i$ and $\mu = \sum_{i=1}^T p_i = \mathbb{E}[X]$. Then, for all $\Lambda > 0$,*

$$\Pr[X \geq (1 + \Lambda)\mu] \leq \exp\left(-\frac{\Lambda^2}{2 + \Lambda}\mu\right) \text{ and } \Pr[X \leq (1 - \Lambda)\mu] \leq \exp\left(-\frac{\Lambda^2}{2 + \Lambda}\mu\right).$$

Definition 8. [MU05, Chapter 12] A sequence of random variables X_0, X_1, \dots is a martingale with respect to sequence Y_0, Y_1, \dots , if, for all $n \geq 0$, (1) X_n is a function of Y_0, \dots, Y_n , (2) $\mathbb{E}[|X_n|] < \infty$, and (3) $\mathbb{E}[X_{n+1}|Y_0, \dots, Y_n] = X_n$.

Theorem 20. [McD98, Theorem 3.15] Let X_0, X_1, \dots be a martingale with respect to the sequence Y_0, Y_1, \dots . For $n \geq 0$, let $V = \sum_{i=1}^n \text{var}(X_i - X_{i-1} | Y_0, \dots, Y_{i-1})$ and $b = \max_{1 \leq i \leq n} \sup(X_i - X_{i-1} | Y_0, \dots, Y_{i-1})$, where \sup is taken over all possible assignments to Y_0, \dots, Y_{i-1} . Then, for any $t, v \geq 0$,

$$\Pr[(X_n \geq X_0 + t) \wedge (V \leq v)] \leq \exp \left\{ -\frac{t^2}{2v + 2bt/3} \right\}.$$

Fact 2. Suppose that x_1, x_2, \dots, x_n are positive real numbers. Then,

$$\frac{x_1 + x_2 + \dots + x_n}{n} \geq \frac{n}{1/x_1 + 1/x_2 + \dots + 1/x_n}.$$

Theorem 21 (Cassel’s inequality). [WAT55] Let $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$ be two positive n -tuples with $0 < m \leq \frac{a_k}{b_k} \leq M < \infty$ for each $k \in \{1, \dots, n\}$ and constants m, M . Then,

$$\left(\sum_{k=1}^n w_k a_k^2 \right) \left(\sum_{k=1}^n w_k b_k^2 \right) \leq \frac{(M + m)^2}{4mM} \cdot \left(\sum_{k=1}^n w_k a_k b_k \right)^2.$$

Theorem 22. Suppose that x_1, x_2, \dots, x_n are positive real numbers. If $\max_{i \in [n]} x_i < \gamma \cdot \min_{i \in [n]} x_i$, then

$$\frac{x_1 + x_2 + \dots + x_n}{n} \leq \frac{(\gamma + 1)^2}{4\gamma} \cdot \frac{n}{1/x_1 + 1/x_2 + \dots + 1/x_n}.$$

Proof. Let $a_i = \sqrt{n_i}$ and $b_i = 1/\sqrt{n_i}$. $\{a_i/b_i\}$ forms a tuple which is identical to $\{n_i\}$. Since $\max_{i \in [n]} T_i < \gamma \cdot \min_{i \in [n]} T_i$, there exist appropriate constants m and $M = \gamma \cdot m$ that satisfies the condition in Theorem 21. Let $w_k = 1$ for all k , then

$$\left(\sum_{k=1}^n n_k \right) \left(\sum_{k=1}^n \frac{1}{n_k} \right) = \left(\sum_{k=1}^n w_k a_k^2 \right) \left(\sum_{k=1}^n w_k b_k^2 \right) \leq \frac{(M + m)^2}{4mM} \cdot \left(\sum_{k=1}^n w_k a_k b_k \right)^2 = \frac{(\gamma + 1)^2}{4\gamma} \cdot n^2.$$

By rearranging we obtain the desired inequality. □

B Preliminaries (cont’d)

B.1 Model

We present our protocol analysis in the bounded-delay network, as in [GKL20]. The model extends the static setting model with a fixed number of parties used in [GKL14, GKL15] for the analysis of the Bitcoin backbone protocol, to the dynamic setting with a varying number of parties. These models are in turn based on Canetti’s formulation of “real world” notion of protocol execution [Can00a, Can00b, Can01] for multi-party protocols. We describe the parts that are common to these two models and highlight the differences.

Round structure and protocol execution. As in previous formulations, which we follow almost *verbatim*, we let the protocol execution proceed in rounds (note these are not message passing rounds). An environment program denoted \mathcal{Z} provides inputs to parties that execute the protocol Π . Our adversarial model is both *adaptive* and *rushing*. *Adaptive* means that the adversary \mathcal{A} is allowed to take control of parties on the fly; while *rushing* means that \mathcal{A} gets to see all honest parties’ messages before deciding his strategy in any given round. The parties’ communication mechanism is captured by a diffusion (“gossiping”) functionality. It allows the order of messages to be controlled by \mathcal{A} , i.e., there is no atomicity guarantees in message broadcast [HT94], and, furthermore, the

adversary is allowed to “spooﬀ” the source information on every message (i.e., communication is not authenticated). Still, the adversary cannot change the contents of the message nor prevent them from being delivered (note that in the bounded delay model, honest parties’ message delay has an upper bound of Δ rounds).

A *system of interactive Turing machines* (ITM’s) in the sense of [Can00b] is formed by a pair of (\mathcal{Z}, C) where C denotes a control program encoding the parties that *may* become active in a protocol execution (parties come from a universe \mathcal{U} of parties), and \mathcal{Z} denotes an environment program that interacts with other instances of programs that it spawns at the discretion of the control program C . An execution is w.r.t. a program Π , an adversary \mathcal{A} (which is another ITM) and the universe of parties \mathcal{U} . In addition, C maintains a flag for each instance of an ITM (abbreviated ITI in the terminology of [Can00b]), that is called the **ready** flag and is initially set to false for all parties. Observe that parties are unaware of the set of activated parties.

The joint hash function/network functionality. In order to ensure “fairness” to all miners, we present the dual functionality that is available to all parties running the protocol and the adversary and abstracts the hash function and the network.

- *The hash function functionality.* We adopt a joint random oracle [BR93] functionality to capture the parties’ access to the hash function. It accepts queries of the form (compute, x) and (verify, x, y). For the first type of query, assuming x was never queried before, a value y is sampled from $\{0, 1\}^\kappa$ and it is entered to a table T_H . If x was queried before the pair (x, y) is recovered from T_H . The value y is provided as an answer. For the second type of query, a membership test is performed on the table. In each round, each honest party P_i is allowed to ask q queries, and is given unlimited queries for “verification” for the function $H(\cdot)$. On the other hand, \mathcal{A} is given a number of queries that cannot exceed $t_r \cdot q$ the upper bound on the number of corrupted parties that may be activated in round r ; no verification queries are provided to \mathcal{A} (the adversary can easily simulate those locally). The bound for the adversary is determined as follows. Whenever a corrupted party is activated the bound is increased by q ; whenever a query is asked the bound is decreased by 1 (it is not necessary that the specific corrupted party makes the query). The value q is a polynomial function of κ in synchronous model but 1 in bounded delay network.
- *The diffusion functionality.* Message passing and round bookkeeping is maintained by this functionality. A round variable *round* is initialized to 0. For each party a string denoted by `RECEIVE()` is maintained and the party is allowed to fetch the contents of its corresponding `RECEIVE()` at any time. The functionality records all messages of the form (`Diffuse`, m) it receives from the parties. Completion of a round for a party is indicated by sending a special message (`RoundComplete`). The adversary \mathcal{A} is allowed to receive all the currently recorded `Diffuse` messages at any time and messages to the `RECEIVE()` strings as desired. The round is completed when the adversary submits its (`RoundComplete`) message. In such case, the functionality inspects the contents of all `RECEIVE()` strings and includes any messages m that were diffused by the parties but not contributed by the adversary to the `RECEIVE()` tapes (in the bounded-delay network, the adversary includes such messages Δ rounds ago thus guaranteeing message delivery up to Δ rounds). It also flushes any diffuse records that are placed in the `RECEIVE()` string of all parties. The variable *round* is then incremented and a new round begins.

The dynamic setting. Given the functionalities as described above, the adversary can choose the termination of the round thus deciding on the spot how many honest parties were activated adaptively. In each round, the number of parties that are active in the protocol is denoted by n_r and is equal to the total number of parties that have submitted the (`RoundComplete`) indicator to the diffusion functionality and have their internal flag `ready` set to true. Determining n_r can only be done by examining the view of all honest parties and is not a quantity that is accessible to any

of the honest parties individually. The number of corrupt parties controlled by \mathcal{A} in a round r is similarly denoted by t_r . Note that the corrupt parties come from two sources—they are activated as corrupted or adaptively corrupted (at any time)—and once an honest party is corrupted, it is considered as controlled by the adversary in all subsequent rounds (if activated).

Parties, when activated, are able to read their input tape $\text{INPUT}()$ and communication tape $\text{RECEIVE}()$ from the diffusion functionality. If a party finds that its ready flag is false, it enters a “bootstrapping” mode where it will diffuse a discovery message and synchronize (in the case of Nakamoto consensus, the party will send a request for the latest blockchains, will collect all of them until a time-out parameter is reached and then will pick the most difficult one to start mining). When the synchronization phase terminates, the party will set its `ready` flag to true and after this point it will be counted among the honest parties. An honest party goes “offline” when it misses a round, i.e., the adversary issues a `(RoundComplete)` but that party misses the opportunity to complete its computation. To record this action, whenever this happens we assume that the party’s ready flag is set to false (in particular this means that a party is aware that it went offline; note, however, that the party does not need to report it to anyone). Also observe that parties are unaware of the set of activated parties. As in previous works, we assume, without loss of generality, that each honest party has the same computational power.

The protocol class that we will analyze will not be able to preserve its properties for arbitrary sequences of parties. Thus, we restrict the way the number of parties fluctuate *both* over a long period of time and over a short period of time.

Definition 1. For $\gamma, \Gamma \in \mathbb{R}^+$, we call a sequence $(n_r)_{r \in \mathbb{N}}$ $(\langle \gamma, \sigma \rangle, \langle \Gamma, \Sigma \rangle)$ -*respecting* if it holds that in a sequence of rounds S with $|S| \leq \Sigma$ rounds, $\max_{r \in S} n_r \leq \Gamma \cdot \min_{r \in S} n_r$ and for any consecutive sub-sequence rounds $S' \preceq S$ with $|S'| \leq \sigma$ rounds, $\max_{r \in S'} n_r \leq \gamma \cdot \min_{r \in S'} n_r$.

Definition 1 is consistent with the notion introduced in [GKL17], except that there the party fluctuation is expressed with respect to the number of honest parties. Note that the two fluctuations are related by a constant (concretely, $2 - \delta$; see Table 1). Yet, it turns out that expressing statements in terms of honest-party fluctuation will considerably simplify them. Thus, in a few cases we will slightly overload notation and keep using γ, Γ , while we are actually referring to honest-party fluctuation. Furthermore, note the addition to the definition of the parameters for short-term fluctuation.

An environment that is $(\langle \gamma, \sigma \rangle, \langle \Gamma, \Sigma \rangle)$ -respecting should satisfy $\gamma^{\lfloor \Sigma/\sigma \rfloor} \geq \Gamma$, i.e., the accumulation of short-term fluctuation values could surpass the long-term fluctuation value. Further, note that although the sequence can capture exponential growth, the total run time is bounded by a polynomial (in κ), and thus the Σ/σ ratio is also polynomially bounded.

The term $\{\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}^P(z)\}_{z \in \{0,1\}^*}$ denotes the random variable ensemble describing the view of party P after the completion for an execution running protocol Π with environment \mathcal{Z} and adversary \mathcal{A} , on input $z \in \{0,1\}^*$. We will consider a “standalone” execution without any auxiliary information, and restrict ourselves to executions with $z = 1^\kappa$; thus, we will simply refer to the ensemble by $\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}^P$.

Properties of protocols. In our theorems we will be concerned with *properties* of protocols Π running in the above setting. Such properties will be defined as predicates over the random variable $\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}$ by quantifying over all possible adversaries \mathcal{A} and environments \mathcal{Z} . Note that all our protocols will only satisfy properties with a small probability of error in κ as well as in a parameter k that is selected from $\{1, \dots, \kappa\}$ (note that in practice one may choose k to be much smaller than κ , e.g., $k = 6$).

B.2 Blockchain Notation

We introduce the blockchain notations that we employ for Bitcoin Cash. Due to the hard fork, Bitcoin Cash’s blockchain does not differ much from Bitcoin’s chain. We use similar notation to that in [GKL17]. Let $G(\cdot)$ and $H(\cdot)$ denote cryptographic hash functions with output in $\{0, 1\}^\kappa$. A block with target $T \in \mathbb{N}$ is a quadruple of the form $B = \langle r, st, x, ctr \rangle$ where $st \in \{0, 1\}^\kappa$, $x \in \{0, 1\}^*$, and $r, ctr \in \mathbb{N}$. The quadruple satisfies the predicate $\text{validblock}_q^T(B)$ defined as

$$(H(ctr, G(r, st, x)) < T) \wedge (ctr \leq q).$$

In real-world situation, as showed above, the parameter $q \in \mathbb{N}$ is a bound on the size of register ctr ; in our treatment we remove this bound and allow ctr to be arbitrary. Instead, we use q to denote the maximum allowed number of hash queries in a round. We do this for convenience and our analysis applies in a straightforward manner to the case that ctr is restricted to the range $0 \leq ctr < 2^{32}$ and q is independent of ctr .

A *blockchain*, or simply a *chain* is a sequence of *blocks*. We call the rightmost block the *head* of the chain, denoted by $\text{head}(\mathcal{C})$. Note that the empty string ε is also a chain, and by convention we set $\text{head}(\varepsilon) = \varepsilon$. Parties that would like to extend a chain \mathcal{C} with $\text{head}(\mathcal{C}) = \langle r, st, x, ctr \rangle$ should compute a valid block $B = \langle r', st', x', ctr' \rangle$ that satisfies $st' = H(ctr, G(r, st, x))$ and $r' > r$, where r' is called the *timestamp* of block B . In case $\mathcal{C} = \varepsilon$, by convention any valid block of the form $\langle r', st', x', ctr' \rangle$ may extend it. In either case we get an extended chain $\mathcal{C}_{\text{new}} = \mathcal{C}B$ that satisfies $\text{head}(\mathcal{C}_{\text{new}}) = B$.

The *length* of a chain $\text{len}(\mathcal{C})$ is its number of blocks. Consider a chain \mathcal{C} of length ℓ and any non-negative integer k . We denote by $\mathcal{C}^{\lceil k}$ the chain resulting from “pruning” the k rightmost blocks. Note that for $k > \text{len}(\mathcal{C})$, $\mathcal{C}^{\lceil k} = \varepsilon$. If \mathcal{C}_1 is a prefix of \mathcal{C}_2 we write $\mathcal{C}_1 \preceq \mathcal{C}_2$.

Given a chain \mathcal{C} of $\text{len}(\mathcal{C}) = \ell$, we let $\mathbf{x}_{\mathcal{C}}$ denote the vector of ℓ values that is stored in \mathcal{C} and starts with the value of the first block. Similarly, $\mathbf{r}_{\mathcal{C}}$ is the vector that contains the timestamps of the blockchain \mathcal{C} .

For a chain of variable difficulty, the target T is recalculated for each block based on the round timestamps of the previous blocks. Specifically, there is a function $D : \mathbb{Z}^* \rightarrow \mathbb{R}$ which receives an arbitrary vector of round timestamps and produces the next target. The value $D(\varepsilon)$ is the initial target of the system. The difficulty of each block is measured in terms of how many times the block is harder to obtain than a block of target T_0 . To be precise, the difficulty of a block with target T will be equal to T_0/T (We denote the difficulty as $1/T$ for simplicity in the full analysis section). We will use $\text{diff}(\mathcal{C})$ to denote the difficulty of a chain. This is equal to the sum of the difficulties of all the blocks that comprise the chain.

B.3 Blockchain Properties

We define two properties of blockchain that it will establish. They are related to the protocol application properties of Consistency and Liveness (cf. Section 4).

The *common prefix* property, parameterized by a value $k \in \mathbb{N}$, considers an arbitrary environment and adversary, and it holds as long as any two parties’ chains are different only in their most recent k blocks. It is actually helpful to define the property between an honest party’s chain and another chain that may be adversarial. The definition is as follows.

Definition 9 (Common Prefix Property). The *common prefix* property Q_{cp} with parameter $k \in \mathbb{N}$ states that, at any round of the execution, if a chain \mathcal{C} belongs to an honest party, then for any valid chain \mathcal{C}' in the same round such that either $\text{diff}(\mathcal{C}') > \text{diff}(\mathcal{C})$, or $\text{diff}(\mathcal{C}') = \text{diff}(\mathcal{C})$ and $\text{head}(\mathcal{C}')$ was computed no later than $\text{head}(\mathcal{C})$, it holds that $\mathcal{C}^{\lceil k} \preceq \mathcal{C}'$ and $\mathcal{C}'^{\lceil k} \preceq \mathcal{C}$.

The second property, called *chain quality*, expresses the number of honest-party contributions that are contained in a sufficiently long and continuous part of a party’s chain. Because we consider chains of variable difficulty it is more convenient to think of parties’ contributions in terms of the total difficulty they add to the chain as opposed to the number of blocks they add (as done in [GKL14]). The property states that adversarial parties are bounded in the amount of difficulty they can contribute to any sufficiently long segment of the chain.

Definition 10 (Chain Quality Property). The chain quality property Q_{cq} with parameters $\mu \in \mathbb{R}$ and $\ell \in \mathbb{N}$ states that for any party P with chain \mathcal{C} in $\text{VIEW}_{\Pi, \mathcal{A}, \mathcal{Z}}$, and any segment of that chain of difficulty d such that the timestamp of the first block of the segment is at least ℓ smaller than the timestamp of the last block, the blocks the adversary has contributed in the segment have a total difficulty that is at most $\mu \cdot d$.

B.4 Ledger Properties

The (main) application of the protocol is a *robust transaction ledger* (cf. [GKL15]), aimed at maintaining a ledger \mathcal{L} of a serialized transaction sequence organized in the form of a blockchain. We now introduce the distinction between \mathcal{L} and $\tilde{\mathcal{L}}$, where \mathcal{L} denotes the settled ledger in the view of the party, and $\tilde{\mathcal{L}}$ denotes the settled ledger with a sequence of transactions appended that are still not settled in the view of the party. Note that it always holds that $\mathcal{L} \preceq \tilde{\mathcal{L}}$. The properties that the protocol application must satisfy are as follows:

- **Consistency:** For any two honest parties P_1, P_2 , reporting $\mathcal{L}_1, \mathcal{L}_2$ at rounds $r_1 \leq r_2$, resp., it holds that \mathcal{L}_1 is a prefix of $\tilde{\mathcal{L}}_2$.
- **Liveness** (Parameterized by $u \in \mathbb{N}$, the “wait time” parameter): If a transaction tx is provided to all honest parties for u consecutive rounds, then it holds that for any player P , tx will be in \mathcal{L} .

C Proof That All Executions Are Typical

Proof of Theorem 8: Since the length of the execution, L , is fixed we will prove the stated bound for a fixed set of consecutive rounds S —or, with respect to the adversary, a fixed set of consecutive queries J —and then apply a union bound over all such sets in the length of the execution. Furthermore, we may assume $|S| \leq \Sigma = \Theta(m)$. Note that $\ell < \Sigma/2$ and we may partition S into parts with size between ℓ and Σ . In the end, we sum over all parts to obtain the desired bound.

Before we consider the random variables, we first propose some useful inequalities that hold for any round r : $[1 - f(T_r^{\max}, h_r)] \leq \mathbf{E}[Y_r | \mathcal{E}_{r-1} = E_{r-1}] \leq \mathbf{E}[D_r | \mathcal{E}_{r-1} = E_{r-1}] = ph_r$, $\mathbf{E}[Y_r^2 | \mathcal{E}_{r-1} = E_{r-1}] \leq ph_r/T_r^{\min}$, and $\mathbf{var}[D_r | \mathcal{E}_{r-1} = E_{r-1}] \leq ph_r/T_r^{\min}$. Let us drop the subscript r for convenience. Suppose that the h honest parties at round r query for targets T_1, \dots, T_h . Observe that all these variables are determined by E_{r-1} . We have

$$\begin{aligned} \mathbf{E}[Y_r | \mathcal{E}_{r-1} = E_{r-1}] &= \sum_{i \in [h]} \frac{1}{T_i} \cdot \frac{T_i}{2^\kappa} \prod_{i < j} [1 - f(T_j, 1)] \geq \sum_{i \in [h]} p \prod_{j \in [h]} [1 - f(T_j, 1)] \\ &\geq \sum_{i \in [h]} p \prod_{j \in [h]} [1 - f(T^{\max}, 1)] = \sum_{i \in [h]} p [1 - f(T^{\max}, h)] = ph [1 - f(T^{\max}, h)], \end{aligned}$$

where the third inequality holds because $f(T, h)$ is increasing in T . For the upper bound on variance,

$$\mathbf{var}[D_r | \mathcal{E}_{r-1} = E_{r-1}] \leq \sum_{i \in [h]} \frac{1}{T_i^2} \cdot \frac{T_i}{2^\kappa} = \sum_{i \in [h]} \frac{p}{T_i} \leq \frac{ph}{T^{\min}}$$

and $\mathbf{E}[Y_r^2 | \mathcal{E}_{r-1} = E_{r-1}]$ is bounded like.

For starter, we fix an execution E_0 just before the beginning of S (or J). We will prove that the statements fail with exponentially small probability for an arbitrary E_0 . Note that E_0 determines the number of parties h_0 and t_0 at the beginning of S (or J) and the target $T(J)$ associated with the first query in J .

For each round $i \in S$, we define a Boolean random variable F_i equal to 1 exactly when all h_i hash values that were returned to the queries of the honest parties were above $\min\{T : f(T, h_i) \geq 2\gamma\Gamma^3 f\}$; define $Z_i = Y_i \cdot F_{i+1} \cdots F_{i+\Delta-1}$. Let G denote the event that the rounds in S are good. Given G , for any $i \in S$, $(F_i = 1) \implies (D_i = 0)$ and so $Q_i \geq Z_i$. Thus, for any d ,

$$\Pr[G \wedge Q(S) \leq d] \leq \Pr[G \wedge Z(S) \leq d],$$

and we may focus on the right-hand side. Identify S with $\{1, \dots, |S|\}$ and partition it with sets of the form $S_j = \{j, j + \Delta, j + 2\Delta, \dots\}$ for $j \in \{0, 1, \dots, \Delta - 1\}$. We will show that, for each part S_j ,

$$\Pr[G \wedge Z(S_j) \leq (1 - \epsilon)[1 - 2\gamma\Gamma^3 f]^\Delta ph(S_j)] \leq e^{-\varphi}.$$

Let us fix a set $S_j = \{s_1, s_2, \dots, s_\nu\}$ with $\nu \geq \lfloor |S|/\Delta \rfloor$, and define the event G_t as the conjunction of the events G and $t = (1 - 2\gamma\Gamma^3 f)^\Delta ph(S_j)$. Note that $|S_j| \leq L$ and so t ranges over a discrete set of size at most L and we can afford a union bound on it. Thus, it is sufficient to show that for any such t ,

$$\Pr[G_t \wedge Z(S_j) \leq [1 - 2\gamma\Gamma^3 f]^\Delta ph(S_j) - t] \leq e^{-\varphi}.$$

To that end, consider the sequence of random variables

$$X_0 = 0; X_k = \sum_{i \in [k]} Z_{s_i} - \sum_{i \in [k]} \mathbf{E}[Z_{s_i} | \mathcal{E}_{s_i-1}], k \in [\nu].$$

This is a martingale with respect to sequence $\mathcal{E}_{s_1-1}(\mathcal{E}_0 = E_0), \dots, \mathcal{E}_{s_\nu-1}, \mathcal{E}$, because $\mathbf{E}[X_k | \mathcal{E}_{s_k-1}] = \mathbf{E}[Z_{s_k} - \mathbf{E}[Z_{s_k} | \mathcal{E}_{s_k-1}] | \mathcal{E}_{s_k-1}] + \mathbf{E}[X_{k-1} | \mathcal{E}_{s_k-1}] = X_{k-1}$ (recall basic properties of conditional expectation [McD98]).

Specifically, the above follows from linearity of conditional expectation and the fact that X_{k-1} is a deterministic function of $\mathcal{E}_{s_{k-1}+\Delta-1} = \mathcal{E}_{s_k-1}$. Furthermore, given an execution E satisfying G_t ,

$$\epsilon \sum_{i \in S_j} \mathbf{E}[Z_i | \mathcal{E}_{s_k-1} = E_{s_k-1}] \geq \epsilon \sum_{i \in S_j} [1 - 2\gamma\Gamma^3 f]^\Delta ph_i = t.$$

Thus, our goal is to show $\Pr[-X_\nu \geq t \wedge G_t] \leq e^{-\varphi}$.

We now provide the details relevant to Theorem 20. Consider an execution E satisfying G_t and let B denote the event $\mathcal{E}_{s_k-1} = E_{s_k-1}$. Note that $Z_{s_k}^2 = Y_{s_k}^2 \cdot F_{s_k+1} \cdots F_{s_k+\Delta-1}$ and all these random variables are independent given B . Since $X_k - X_{k-1} = Z_{s_k} - \mathbf{E}[Z_{s_k} | \mathcal{E}_{s_k-1}]$ and

$$Z_{s_k} - \mathbf{E}[Z_{s_k} | B] \leq \frac{1}{T_{s_k}^{\min}} \cdot \frac{ph_{s_k}}{ph_{s_k}} \leq \frac{\Gamma ph(S_j)}{ph_{s_k} T_{s_k}^{\min} |S_j|} \leq \frac{\Gamma ph(S_j)}{\nu f / (2(2 - \delta)\gamma\Gamma^3)} \leq \frac{2(2 - \delta)\gamma\Gamma^4 t}{\epsilon(1 - 2\gamma\Gamma^3 f)^\Delta f \nu} \stackrel{\text{def}}{=} b,$$

we see that the event G implies $X_k - X_{k-1} \leq b$. With respect to $V = \sum_k \mathbf{var}[X_k - X_{k-1} | \mathcal{E}_{s_{k-1}}] \leq \sum_k \mathbf{E}[Z_{s_k}^2 | \mathcal{E}_{s_{k-1}}]$, using the independence of the random variables and inequalities showed at the beginning of the proof,

$$\sum_{k \in [\nu]} [Z_{s_k}^2 | B] \leq (1 - 2\gamma\Gamma^3 f)^{\Delta-1} \sum_{k \in [\nu]} \frac{(ph_{s_k})^2}{ph_{s_k} T_{s_k}^{\min}} \leq \frac{(1 - 2\gamma\Gamma^3 f)^{\Delta-1}}{f/[2(2-\delta)\gamma\Gamma^3]} \cdot \sum_{k \in [\nu]} (ph_{s_k})^2.$$

Applying Fact 1(b) on this bound, we see that event G_t implies

$$V \leq \frac{2(2-\delta)\gamma\Gamma^4[1 - 2\gamma\Gamma^3 f]^{\Delta-1}}{f|S_j|} \cdot \left(\sum_{k \in [\nu]} ph_{s_k} \right)^2 \leq \frac{2(2-\delta)\gamma\Gamma^4 t^2}{\epsilon^2 f (1 - 2\gamma\Gamma^3 f)^{\Delta+1} \nu} \stackrel{\text{def}}{=} b.$$

In view of these bounds (note that $bt < \epsilon v$), by Theorem 20,

$$\Pr[-X_\nu \geq t \wedge G_t] \leq \exp\left\{-\frac{t^2}{2\nu(1 + \frac{\epsilon}{3})}\right\} \leq \exp\left\{-\frac{\epsilon^2 f [1 - 2\gamma\Gamma^3 f]^{\Delta+1} \nu}{2(2-\delta)\gamma\Gamma^4(1 + \frac{\epsilon}{3})}\right\} \leq e^{-\varphi}, \quad (4)$$

where for the last inequality we used the condition on ℓ (recall that $\nu \geq \ell/\Delta$).

For the bound on $D(S)$, it will be convenient to work per query. Let J denote the queries in S , $\nu = |J|$, and Z_i the difficulty of any block obtained from query $i \in J$. Define the martingale sequence

$$X_0 = 0; \quad X_k = \sum_{i \in [k]} Z_i + \sum_{i \in [k]} \mathbf{E}[Z_i | \mathcal{E}_{i-1}], \quad k \in [\nu].$$

With similar calculations above we obtain that G_t (with $t = \epsilon p \nu$) implies

$$X_k - X_{k-1} \leq \frac{2\gamma\Gamma^2 t}{\epsilon f |S|} \stackrel{\text{def}}{=} b \quad \text{and} \quad V \leq \frac{2\gamma\Gamma^2 t^2}{\epsilon^2 f |S|} \stackrel{\text{def}}{=} v.$$

Applying Theorem 20 we obtain

$$\Pr[X_\nu \geq t \wedge G_t] \leq \exp\left\{-\frac{\epsilon t}{2b(1 + \frac{\epsilon}{3})}\right\} \leq e^{-\varphi}. \quad (5)$$

We next focus on part (b). For each $j \in J$, let A_j be equal to the difficulty of the block obtained with the j -th query as long as the target was at least $T(J)/\tau$; thus $A(J) = \sum_{j \in J} A_j$. If $|J| = \nu$, identify J with $[\nu]$ and define the martingale

$$X_0 = 0; \quad X_k = \sum_{j \in [k]} A_j - \sum_{j \in [k]} \mathbf{E}[A_j | \mathcal{E}_{j-1}], \quad k \in [\nu].$$

For all $k \in [\nu]$ we have $X_k - X_{k-1} \leq \tau/T(J)$, $\mathbf{var}[X_k - X_{k-1} | \mathcal{E}_{k-1}] \leq p\tau/T(J)$, and $\mathbf{E}[A_j | \mathcal{E}_{j-1}] \leq p$. We may apply Theorem 20 with $b = \tau/T(J)$, $v = bp\nu \leq bt/\epsilon$, and $t = \max\{\epsilon p \nu, 2(\frac{1}{\epsilon} + \frac{1}{3})b\varphi\}$. We obtain

$$\Pr\left[\sum_{j \in J} A_j \geq p\nu + t\right] \leq \exp\left\{-\frac{t}{2b(\frac{1}{3} + \frac{1}{\epsilon})}\right\} \leq e^{-\varphi}. \quad (6)$$

Based on the three martingale inequalities (4)(5)(6), the probability of one partition S is not typical is bounded by $1 - (1 - e^{-\varphi})^3 \leq 3e^{-\varphi}$. For a system that runs for L steps, the total number of partitions is $\binom{L}{2}$. Thus, the probability for the entire execution is not typical is bounded by $3\binom{L}{2}e^{-\varphi}$.

For part (c) and $i \in \{0, 1, 2, 3\}$, let $B_i = \langle r_i, st_i, x_i, ctr_i \rangle$ and $g_i = G(r_i, st_i, x_i)$. If a block extends two distinct blocks, then a collision has occurred. To see this, suppose block B_3 extends two distinct blocks B_1 and B_2 , then $st_3 = H(ctr_1, g_1) = H(ctr_2, g_2)$; implying a collision either in H or in G , since B_1 and B_2 are distinct. The existence of an insertion or a copy implies a collision as well. If the total running time of the system of ITM's is L then it holds that there are at most L queries posed to G, H . It follows that the probability of a collision occurring is $\binom{L}{2} 2^{-\kappa+1} \leq 2^{-\kappa+1+2\log L}$. Note that, for polynomially many rounds in κ , the probability that a guessed block occurs is exponentially small in κ .

Recall that φ is a polylogarithmic function of κ (e.g., $\varphi = O(\log^2 \kappa)$). Combining probabilities on both martingale and collision, the overall error probability is $\text{poly}(L) \cdot e^{-\Omega(\text{polylog}(\kappa))}$.

D The SMA Target Recalculation Function (cont'd)

D.1 SMA: Full Analysis

In this section we present the analysis of the Bitcoin Cash protocol with SMA function. In order to make this section self-contained, we state all the preliminaries. However, we omit some of the proofs which generally follow those in Section 4.2.

For convenience, Table 2 summarizes the set of parameters introduced so far. Our security parameter is κ , and note that $\varphi = \Theta(m) = \text{polylog}(\kappa)$.

Table 2: *Summary of parameters (SMA).*

- | |
|--|
| <ul style="list-style-type: none"> – δ: Advantage of honest parties, $\forall r(t_r/h_r < 1 - \delta)$. – $\gamma, \sigma, \Gamma, \Sigma$: Determines how the number of parties fluctuates across rounds in a period, cf. Definition 1 and Fact 1. – f: Probability at least one honest party succeeds in a round assuming h_0 parties and target T_0 (the protocol's initialization parameters). – m: the length of an epoch in number of blocks. – τ: The dampening filter, cf. Definition 7. – λ: The ratio of target range, with relation to target recalculation algorithm. – ϵ: Quality of concentration of random variables, cf. Definition 14. – κ: The length of the hash function output. – φ: Related to the properties of the protocol. – L: The total number of rounds in the execution. |
|--|

Our probability space is over all executions of length at most some polynomial in κ . We will denote by \mathbf{Pr} the probability measure of this space. We also define the random variable \mathcal{E} taking values on this space and with a distribution induced by the random coins of all entities (adversary, environment, parties) and the random oracle.

Recall that in our modeling of a bounded-delay network, each party's query bound to the random oracle (RO) per round is $q = 1$. Now suppose at round r exactly n parties query the RO with a target T . Then the probability that at least one of them will succeed is

$$f(T, h) = 1 - (1 - pT)^h \leq phT, \text{ where } p = 1/2^\kappa.$$

As in prior work, we denote $f_0 = f(T_0, h_0)$ where T_0 and h_0 are the initial target and estimate of number of honest parties. The objective of the target recalculation mechanism would be to

maintain a target T for each party such that $f(T, h_r) \approx f_0$ for all rounds r . For simplicity, we will drop the subscript from f_0 , and will always specify the two arguments of $f(\cdot, \cdot)$ to avoid confusion.

During a round r of an execution E , the honest parties might be split and work on different chains, and thus might query the RO on different targets. Denote by T_r^{\min} and T_r^{\max} the minimum and maximum of these targets, respectively. We say r is a target-recalculation point of a valid chain \mathcal{C} , if there is a block with timestamp r (recall that the target is adjusted for every block).

Next, and following [GKL17] we define properties *goodness* and *accuracy*, which we will then show most executions satisfy, and which will help achieve the desired application. In addition,

Definition 11 (Goodness). Round r is *good* if $f/2\gamma\Gamma \leq ph_r T_r^{\min}$ and $ph_r T_r^{\max} \leq (1 + \delta)\gamma\Gamma f$. A target-recalculation point r is *good* if the target T for the next block satisfies $f/2\Gamma \leq ph_r T \leq (1 + \delta)\Gamma f$. Finally, a chain is *good* if all its target-recalculation points are good.

Definition 12 (Accuracy). A block created at round u is *accurate* if it has a timestamp v such that $|u - v| \leq \ell + 2\Delta$. A chain is *accurate* if all its blocks are accurate. A chain is *stale*, if for some $u \geq \ell + 2\Delta$ it does not contain an honest block with timestamp $v \geq u - \ell - 2\Delta$.

We already mentioned that an epoch a sequence of m blocks. Let u and v be the corresponding initial and last target-recalculation points. Thus, the *duration* of an epoch is $v - u$.

For a given round r , we let \mathcal{S}_r denote a set of chains that belong, or could potentially belong to an honest party. Being explicit about this set of chains will help expressing a number of predicates (see below). Specifically, \mathcal{S}_r includes:

- Chain \mathcal{C} that belongs to an honest party;
- Chain \mathcal{C} with $\text{diff}(\mathcal{C}) > \text{diff}(\mathcal{C}')$ for some chain \mathcal{C}' of an honest party;
- chain \mathcal{C} with $\text{diff}(\mathcal{C}) = \text{diff}(\mathcal{C}')$ for some chain \mathcal{C}' of an honest party and $\text{head}(\mathcal{C})$ was computed no later than $\text{head}(\mathcal{C}')$.

Note that these chains should exist and be valid at round r . As in [GKL17], we now define the following series of predicates.

Definition 13. For a round r , let:

- $\text{GOODCHAINS}(r) \triangleq$ “For all $u \leq r$, every chain in \mathcal{S}_u is good.”
- $\text{GOODROUNDS}(r) \triangleq$ “All rounds $u \leq r$ are good.”
- $\text{NOSTALECHAINS}(r) \triangleq$ “For all $u \leq r$, there are no stale chains in \mathcal{S}_u .”
- $\text{ACCURATE}(r) \triangleq$ “For all $u \leq r$, all chains in \mathcal{S}_u are accurate.”
- $\text{DURATION}(r) \triangleq$ “For all $u \leq r$ and duration Λ of any epoch in $\mathcal{C} \in \mathcal{S}_u$, $\frac{1}{2(1+\delta)\Gamma^2} \cdot \frac{m}{f} \leq \Lambda \leq 2(1 + \delta)\Gamma^2 \cdot \frac{m}{f}$.”

Random variables. We are interested in estimating the difficulty acquired by honest parties during a sequence of rounds. For a given round r , the following real-valued random variables are defined in [GKL20]:

- D_r : “Sum of the difficulties of all blocks computed by honest parties.”
- Y_r : “Maximum difficulty among all blocks computed by honest parties.”
- Q_r : “Equal to Y_r when $D_u = 0$ for all $r < u < r + \Delta$ and 0 otherwise.”

A round r such that $D_r > 0$ is called *successful* and one where $Q_r > 0$ *isolated successful*.

Regarding the adversary, in order to overcome the fact that he can query the oracle for arbitrarily low targets and thus obtain blocks of arbitrarily high difficulty, we would like to upper-bound the difficulty he can acquire during a set J of queries. This is achieved by associating a set of consecutive

adversarial queries J with the target of its first query. We denote this target $T(J)$, and say that $T(J)$ is *associated* with J . We then define $A(J)$ and $B(J)$ to be equal to the sum of the difficulties of all blocks computed by the adversary during queries in J for target at least $T(J)/\lambda\Gamma$ and $T(J)$, respectively – i.e., queries in J for targets less than $T(J)/\lambda\Gamma$ (resp. $T(J)$) do not contribute to $A(J)$ (resp. $B(J)$).

For simplicity, we write $h(S) = \sum_{r \in S} h_r$ for a set of rounds S and queries J (similarly, $t(S)$, $D(S)$, $Y(S)$, $Q(S)$, $A(J)$ and $B(J)$).

Regarding protocol executions, let \mathcal{E}_{r-1} fix the execution just before round r . In particular, a value E_{r-1} of \mathcal{E}_{r-1} determines the adversarial strategy and so determines the targets against which every party will query the oracle at round r and the number of parties h_r and t_r , but it does not determine D_r or Q_r . For an adversarial query j we will write E_{j-1} for the execution just before this query.

Furthermore, we are interested in the range of targets during an execution in a $(\langle \gamma, \sigma \rangle, \langle \Gamma, \Sigma \rangle)$ -respecting environment. Intuitively, the range of target T is related to the number of parties \mathbf{n} . It is not hard to see that the ratio T_{max}/T_{min} should be larger than that on number of parties, or the protocol would not be able to achieve the goal of maintaining a stable block generating rate ($f \approx pTn$). In addition, recall that according to the target recalculation function $T' = \frac{\Delta}{m^2/f} \cdot \sum_{i \in [m]} T_i$, the targets of two neighboring blocks will not differ by too much, thus forming a relatively “smooth” oscillation target curve. Finally, we define a new variable $\lambda \geq 1$ that helps bound the fluctuation on the targets. Specifically, we assume that the targets during consecutive rounds $|S| \leq \Sigma$ satisfy $T_{max} \leq \lambda\Gamma \cdot T_{min}$.

In order to obtain meaningful concentration of random variables, we have to consider a sufficiently long sequence with a number of rounds at least

$$\ell = \frac{4(1+3\epsilon)}{\epsilon^2 f [1 - (1+\delta)\gamma\Gamma f]^{\Delta+1}} \cdot \max\{\Delta, \lambda\Gamma\} \cdot \gamma\Gamma^2 \cdot \varphi. \quad (7)$$

We will assume that ℓ is appropriately small compared to the duration of an epoch. Specifically,

$$2\ell + 6\Delta \leq \frac{\epsilon m}{2(1+\delta)\Gamma^2 f}. \quad (C3)$$

In addition, we would like the advantage δ of the honest parties over adversarial parties to be large enough to absorb error factors. Thus, we require the following inequalities:

$$\frac{4\lambda\Gamma}{(\lambda\Gamma + 1)^2} \cdot [1 - (1+\delta)\gamma\Gamma f]^\Delta > 1 - \epsilon \text{ and } \epsilon \leq \delta/8 \leq 1/8. \quad (C4)$$

Note that the chain-growth lemma still works here (cf. Lemma 6).

Typical executions. We have introduced the notion of *typical executions* in Section 4.2. Regarding SMA, we note that Bitcoin Cash (as well as other cryptocurrencies) adopts strategies to reduce the impact of bad events to the minimal extent, such as rearranging the blocks temporarily before recalculating the target (as opposed to the Bitcoin implementation, which directly adopts the last block’s timestamp). Since our definition automatically gets rid of these bad events, we will not consider such mechanisms in our analysis.

We are now ready to specify the typical execution associated with SMA algorithm.

Definition 14 (Typical execution). An execution E is *typical* if the following hold:

- (a) For any set S of at least ℓ consecutive good rounds,

$$(1 - \epsilon)[1 - (1 + \delta)\gamma\Gamma f]^\Delta ph(S) < Q(S) \leq D(S) < (1 + \epsilon)ph(S).$$

(b) For any set J indexing a set of consecutive queries of the adversary and $\alpha(J) = 2(\frac{1}{\epsilon} + \frac{1}{3})\varphi/T(J)$,

$$A(J) < p|J| + \max\{\epsilon p|J|, \lambda\Gamma\alpha(J)\} \text{ and } B(J) < p|J| + \max\{\epsilon p|J|, \alpha(J)\}.$$

(c) No insertions, no copies, and no predictions occurred in E .

The next proposition is a simple application of Definition 14 as well as the honest-majority-environment assumption.

Proposition 23. *Let E be a typical execution in a $(\langle\gamma/(2-\delta), \sigma\rangle, \langle\Gamma/(2-\delta), \Sigma\rangle)$ -respecting environment. Let $S = \{r \mid (u \leq r \leq v) \wedge (v - u \geq \ell)\}$ be a set of consecutive good rounds and J the set of adversarial queries in $U = \{r \mid u - \Delta \leq r \leq v + \Delta\}$. The following inequalities hold:*

- (a) $(1 + \epsilon)p|J| \leq Q(S) \leq D(U) < (1 + 5\epsilon)Q(S)$.
- (b) $T(J)A(J) < \epsilon m/4(1 + \delta)$ or $A(J) < (1 + \epsilon)p|J|$ and $\lambda\Gamma \cdot T(J)B(J) < \epsilon m/4(1 + \delta)$ or $B(J) < (1 + \epsilon)p|J|$.
- (c) If w is a good round such that $|w - r| \leq \Sigma$ for any $r \in S$, then $Q(S) > (1 + \epsilon)[1 - (1 + \delta)\gamma\Gamma f]^\Delta |S|ph_w/\Gamma$. If in addition $T(J) \geq T_w^{\min}$, then $A(J) < (1 - \delta + 3\epsilon)Q(S)$.

Proof. We only consider part (b): if $\epsilon p|J| \geq \lambda\Gamma\alpha(J)$, Definition 14(b) applies directly. Otherwise, $p|J| < \lambda\Gamma\alpha(J)/\epsilon$ we get

$$T(J) \cdot A(J) < \frac{2}{\epsilon^2} \cdot (1 + \epsilon)(1 + \frac{\epsilon}{3})\lambda\Gamma\varphi < \frac{f\ell}{2\gamma\Gamma^2} < \frac{\epsilon m}{4(1 + \delta)\gamma\Gamma^2}.$$

This follows Equation 7 and Condition (C3). □

We are now able to show that almost all Bitcoin Cash protocol executions polynomially bounded (in κ) are typical. Formally:

Theorem 24. *Assuming the Bitcoin Cash backbone protocol runs for L rounds, the event “ E is not typical” is bounded by $\text{poly}(L) \cdot e^{-\Omega(\text{polylog}(\kappa))}$.*

Next, we consider the validity of the predicates in Definition 13 over the space of typical execution in a $(\langle\gamma/(2-\delta), \sigma\rangle, \langle\Gamma/(2-\delta), \Sigma\rangle)$ -respecting environment, as well as implications among them. Furthermore, we assume that all the requirements hold for the initialization parameters h_0 and T_0 . (Some of the statements’ proofs require the execution to have a safe start [cf. Remark 1].)

Lemma 25. $\text{GOODROUNDS}(r - 1) \implies \text{NOSTALECHAINS}(r)$.

Proof. For the sake of a contradiction, consider a chain $\mathcal{C} \in \mathcal{S}_r$ which has not been extended by the honest party for at least $\ell + 2\Delta$ rounds. Without loss of generality, let r denote the least round with this property. Let B be the last block of \mathcal{C} computed by the honest party (possibly the genesis block) and let w be its timestamp. Set $S = \{u : w + \Delta \leq u \leq r - \Delta\}$ and $U = \{u : w \leq u \leq r\}$. Therefore, to reach a contradiction it suffices to show that the adversary’s accumulated difficulty $d < Q(S)$.

Let J denote the queries in U starting from the first adversarial query attempting to extend B . We learn from Condition (C3) that targets the adversary can query during U is bounded by $T(J)/\lambda\Gamma$. Thus, $d < A(J)$. If $A(J) < (1 + \epsilon)p|J|$, then $A(J) < Q(S)$ is obtained by Proposition

23(a). Otherwise, $A(J) < (\frac{1}{\epsilon} + 1)\lambda\Gamma\alpha(J) = 2(\frac{1}{\epsilon} + 1)(\frac{1}{\epsilon} + \frac{\epsilon}{3})\lambda\Gamma\varphi/T(J)$. However, by considering only the first ℓ rounds in S , $h(S) \geq h_u\ell/\gamma$. We have

$$\begin{aligned} Q(S) &> (1 - \epsilon)[1 - (1 + \delta)\gamma\Gamma f]^\Delta \cdot \frac{ph_u\ell T(J)}{\gamma T(J)} \\ &> \frac{(1 - \epsilon)[1 - (1 + \delta)\gamma\Gamma f]^\Delta f\ell}{2\gamma\Gamma^2 T(J)} \geq \frac{2(1 - \epsilon)(1 + 3\epsilon)\lambda\Gamma\varphi}{\epsilon^2 T(J)} \geq A(J). \end{aligned}$$

In either situation, we obtain the desired inequality $d < Q(S)$. \square

Corollary 26. $\text{GOODROUNDS}(r - 1) \implies \text{ACCURATE}(r)$.

Lemma 27. $\text{GOODROUNDS}(r - 1) \wedge \text{GOODCHAINS}(r - 1) \implies \text{DURATION}(r)$.

Proof. Assume—towards a contradiction—that $\text{DURATION}(r)$ is false. Without loss of generality, assume $w \leq r$ associated with a chain $\mathcal{C} \in \mathcal{S}_w$ and duration Λ is the smallest round that does not satisfy

$$\frac{1}{2(1 + \delta)\Gamma^2} \cdot \frac{m}{f} \leq \Lambda \leq 2(1 + \delta)\Gamma^2 \cdot \frac{m}{f}.$$

For the upper bound, we will show that the honest parties can by themselves obtain m blocks. Note that Lemma 25 implies that two honest blocks in this epoch exist with timestamps u and v that $v - u \geq \Lambda - 2\ell + \Delta$. We define $S = \{r : u + \Delta \leq r \leq v + \Delta\}$ the time that the honest parties can contribute to the difficulty. Assuming $\Lambda > 2(1 + \delta)\Gamma^2 m/f$, Condition (C3) implies that $|S| \geq \Lambda - 2\ell - 6\Delta \geq 2(1 + \delta)(1 - \epsilon)\Gamma^2 m/f$. We denote $r^* \in S$ the timestamp of the block with lowest target (i.e., highest difficulty) and T_{r^*} its associated target. Thus we have

$$Q(S) > (1 - \epsilon)[1 - (1 + \delta)\gamma\Gamma f]^\Delta \cdot \frac{f|S|}{2\Gamma^2 T_{r^*}} \geq (1 + \delta)(1 - \epsilon)^3 \cdot \frac{m}{T_{r^*}} > \sum_{r \in [m]} \frac{1}{T_i}.$$

For the first inequality, we used Definition 14(a) and that r^* is a good target-recalculation point; the lower bound for $|S|$ above for the next and Condition (C4) for the last one. Result here contradicts Chain Growth, since the honest parties at round v have already accumulated more than $\sum_{i \in [m]} 1/T_i$ difficulty on top of u .

For the sake of proving the lower bound, we are going to argue that even if the honest parties and the adversary join forces they still cannot obtain m blocks. We denote u, v timestamps of the the first and last block in the epoch, respectively. Then, the honest parties can contribute difficulty during $S = \{u, \dots, v\}$, and queries are available for the adversary during $S' = \{u - \ell - 2\Delta, \dots, v + \ell + 2\Delta\}$. Note that S' contains all rounds that the adversary can contribute, since we have Corollary 26 and then $\text{ACCURATE}(r)$ holds. Similarly, we denote $r^* \in S'$ the timestamp of the block with highest target (i.e., lowest difficulty) and T_{r^*} its associated target. We claim that the adversary start with the first query for target T_{r^*} (so that $T(J) = T_{r^*}$). Since r^* is a good target-recalculation point, it follows Fact 1(a) that $ph(S) \leq p\Gamma h_{r^*} |S| \leq (1 + \delta)\Gamma^2 f |S| / T_{r^*}$. Thus,

$$D(S) < (1 + \epsilon)ph(S) \leq (1 + \epsilon)(1 + \delta) \cdot \frac{\Gamma^2 f |S|}{T_{r^*}} \leq (1 + \epsilon) \cdot \frac{m}{2T_{r^*}}.$$

With respect too the adversary, if $\tau T(J)B(J) < \epsilon m/4$, then the total number of blocks is less than m and we are done. Otherwise, by Proposition 23(b) we have

$$\begin{aligned} B(J) &< (1 + \epsilon)p|J| \leq (1 + \epsilon)(1 - \delta)ph(S') \leq (1 + \epsilon)(1 - \delta)p\Gamma h_{r^*} |S'| \\ &\leq (1 - \delta^2)(1 + \epsilon)(1 + \frac{2\ell + 4\Delta}{|S|})|S| \cdot \frac{\Gamma^2 f}{T_{r^*}} \leq (1 - \delta)(1 + \epsilon)^2 \cdot \frac{m}{2T_{r^*}}. \end{aligned}$$

For the last inequality, note that since Condition (C4) we have $\epsilon \leq \delta/4$ and T_{r^*} is the maximum target in the epoch,

$$B(J) + D(S) < \frac{(1 + \epsilon) + (1 - \delta)(1 + \epsilon)^2}{2} \cdot \frac{m}{T_{r^*}} < \frac{m}{T_{r^*}} \leq \sum_{r \in [m]} \frac{1}{T_i},$$

which means the total accumulated difficulty cannot produce m blocks. \square

Lemma 28. $\text{GOODROUNDS}(r - 1) \implies \text{GOODCHAINS}(r)$.

Proof. For starter, we recall our assumption that all the target recalculation points in the first epoch are good. Therefore, it suffices to show that the next target-recalculation point will be good, when given an epoch with m good recalculation points. We denote u, v timestamps of the epoch beginning and ending, respectively (a.k.a. $v = u + \Lambda < r$). Let T' denote the target of the next block, our goal is to show that $f/2\Gamma \leq ph_v T' \leq (1 + \delta)\Gamma f$.

We begin with the lower bound. If $\Lambda \geq \Gamma m/f$, i.e., $T' \geq \Gamma \cdot T^{\text{avg}}$, we pick the block in the epoch with lowest target and denote r^* its timestamp and T^{min} its associated target. Thus, $ph_v T' \geq ph_{r^*} T'/\Gamma \geq ph_{r^*} T^{\text{avg}} \geq ph_{r^*} T^{\text{min}} \geq f/2\Gamma$, because r^* is assumed to be a good target-recalculation point.

Now we assume $\Lambda < \Gamma m/f$, which implies $\Lambda \leq (T'/T^{\text{avg}})(m/f)$. It is clear that all the blocks are computed in either $S = \{u, \dots, v\}$ by the honest parties or $S' = \{u - \ell - 2\Delta, \dots, v + \ell + 2\Delta\}$ by the adversary. Let J denote the set of queries available to the adversary in S' . Note that, by Condition (C3) and Lemma 27, $|S'| = \Lambda + 2\ell + 4\Delta \leq (1 + \epsilon)\Lambda$. We have

$$B(J) < (1 - \delta)(1 + \epsilon)ph(S') \leq (1 - \delta)(1 + \epsilon)p\Gamma h_v |S'| \leq (1 - \delta)(1 + \epsilon)^2 p\Gamma h_v \Lambda.$$

Similarly, $D(S) < (1 + \epsilon)ph(S) \leq (1 + \epsilon)p\Gamma h_v \Lambda$. Assuming $ph_v T' < f/2\Gamma$ and recalling Fact 2, we obtain the contradiction

$$2\Gamma ph_v \Lambda \leq 2\Gamma ph_v \cdot \frac{T'}{T^{\text{avg}}} \cdot \frac{m}{f} < \frac{m}{T^{\text{avg}}} \leq \sum_{i \in [m]} \frac{1}{T_i} \leq D(S) + B(J) < (2 + 4\epsilon - \delta)\Gamma ph_v \Lambda \leq 2\Gamma ph_v \Lambda.$$

Then we move to the upper bound. Note that the adversary can be silent, so here we only consider the honest parties contributing difficulty during $S = \{u + \ell + 3\Delta, \dots, v - \ell - 3\Delta\}$. If $\Lambda \leq m/\Gamma f$, then $T' \leq T^{\text{avg}}/\Gamma$. We pick the block in the epoch with highest target and denote r^* its timestamp and T^{max} its target. Thus, $ph_v T' \leq p\Gamma h_{r^*} T' \leq ph_{r^*} T^{\text{avg}} \leq ph_{r^*} T^{\text{max}} \leq (1 + \delta)\Gamma f$ where r^* is a good target-recalculation point.

Next, we may assume $\Lambda > m/\Gamma f$, which implies $\Lambda \geq (T'/T^{\text{avg}})(m/f)$ and $|S| = |\Lambda - 2\ell - 6\Delta| \geq (1 - \epsilon)\Lambda$. Assuming $ph_v T' > (1 + \delta)\Gamma f$, we obtain the following contradiction

$$\begin{aligned} \frac{ph_v \Lambda}{(1 + \delta)\Gamma} &\geq \frac{ph_v}{(1 + \delta)\Gamma} \cdot \frac{T'}{T^{\text{avg}}} \cdot \frac{m}{f} > \frac{m}{T^{\text{avg}}} \geq \frac{4\lambda\Gamma}{(\lambda\Gamma + 1)^2} \sum_{i \in [m]} \frac{1}{T_i} \\ &\geq \frac{4\lambda\Gamma}{(\lambda\Gamma + 1)^2} \cdot Q(S) > \frac{4\lambda(1 - \epsilon)[1 - (1 + \delta)\gamma\Gamma f]^\Delta}{(\lambda\Gamma + 1)^2} \cdot ph_v |S| \geq \frac{ph_v \Lambda}{(1 + \delta)\Gamma}. \end{aligned}$$

The third inequality follows Theorem 22. For the next one, recall that $\mathcal{C} \in \mathcal{S}_r$ and by Lemma 25 there is a block computed by an honest party among the first and last $\ell + 2\Delta$ rounds of the epoch, respectively. Thus this inequality follows by Chain Growth. The last inequality is a consequence of Condition (C4). \square

Corollary 29. $\text{GOODROUNDS}(r-1) \implies \text{GOODROUNDS}(r)$.

Proof. Consider any $\mathcal{C} \in \mathcal{S}_r$ and let u denote its last recalculation point before r and T the associated target. If r is a recalculation point, it follows directly by Lemma 28. Otherwise, we learn from Lemma 25 that the maximum interval between two blocks is $\ell + 2\Delta$, thus by Fact 1(a) $h_u/\gamma \leq h_r \leq \gamma h_u$ holds. Combining it with $f/2\Gamma \leq ph_u T \leq (1 + \delta)\Gamma f$ we obtain the desired inequality. \square

Theorem 30. *Consider a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, and assume conditions C3 and C4 are satisfied. Then all predicates in Definition 13 hold.*

Theorem 30 follows directly follows from our safe-start assumption (Remark 1).

Blockchain properties. We now show that Bitcoin Cash satisfies the two properties common prefix and chain quality (cf. Section 2), for a suitable respecting environment. First, a preliminary lemma:

Lemma 31. *For any round r of a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment and any two chains \mathcal{C} and \mathcal{C}' in \mathcal{S}_r , the timestamp of $\text{head}(\mathcal{C} \cap \mathcal{C}')$ is at least $r - 2\ell - 4\Delta$.*

Theorem 32 (Common-Prefix). *For a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, the common-prefix property holds for parameter $\epsilon\gamma^2 m/2\Gamma$.*

Proof. Suppose—towards a contradiction—common prefix fails for two chains \mathcal{C}_1 and \mathcal{C}_2 at rounds $r_1 \leq r_2$. It is easy to see that there exist a round $r \leq r_2$ and two chains \mathcal{C} and \mathcal{C}' such that each had at least k blocks after $\text{head}(\mathcal{C} \cap \mathcal{C}')$. In view of Lemma 31, it suffices to show that these blocks were computed within at least $\ell + 2\Delta$ rounds. Suppose the honest parties query during a set of rounds S of size $\ell + 2\Delta$, the accumulated difficulty is $D(S)$, and the number of blocks produced in $|S|$ is at most $T^{\max} \cdot D(S)$ where T^{\max} denote the maximum target among blocks in $|S|$. For any round r^* associated with target T^{\max} , we have

$$T^{\max} \cdot D(S) < (1 + \epsilon)pT^{\max}h(S) \leq (1 + \epsilon)p\gamma T^{\max}h_{r^*}|S| < (1 + \epsilon)\epsilon\gamma^2 m/4\Gamma.$$

For the last inequality, note that by Condition (C3) $|S| < \epsilon m/4(1 + \delta)\Gamma^2 f$ and by Theorem 30 $pT^{\max}h_{r^*} \leq (1 + \delta)\gamma\Gamma f$. By proposition 23 the adversary contributed to less than $(1 + \epsilon)\epsilon\gamma^2 m/2(1 + \delta)\Gamma$ blocks, for a total of less than $\epsilon\gamma^2 m/2\Gamma$. \square

Theorem 33 (Chain-Quality). *For a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, the chain-quality property holds with parameters $\ell + 2\Delta$ and $\mu = \delta - 3\epsilon$.*

Transaction ledger. We conclude this section by showing that a typical execution of the (abstraction of the) Bitcoin Cash protocol we are considering, in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, satisfies the two properties of a robust transaction ledger presented in Section 5. They are the direct consequence of the blockchain properties shown above, following the approach in [GKL17, GKL20].

Theorem 17 (Consistency). *For a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, Consistency is satisfied by setting the settled transactions to be those reported more than $\epsilon\gamma^2 m/2\Gamma$ blocks deep.*

Theorem 18 (Liveness). *For a typical execution in a $(\langle \gamma/(2 - \delta), \ell + 2\Delta \rangle, \langle \Gamma/(2 - \delta), 2(1 + \delta)\Gamma^2 m/f \rangle)$ -respecting environment, Liveness is satisfied for depth $\epsilon\gamma^2 m/2\Gamma$ with wait-time $(\gamma^2 + 1)\epsilon m/f$.*

Proof. We claim that the chain \mathcal{C} of any honest party has at least $\epsilon\gamma^2 m/2\Gamma$ blocks that were computed in the last $\epsilon\gamma^2 m/(1 - 2\epsilon)f + 2\Delta$ rounds. Denote by T^{\min} the lowest target among these blocks and r^* a round associated with this target. If S is its subset without the first and last Δ rounds, by Chain-Growth Lemma, the length of the segment is at least

$$T^{\min} \cdot Q(S) > (1 - \epsilon)[1 - (1 + \delta)\gamma\Gamma f]^\Delta p T^{\min} h(S) \geq \frac{(1 - 2\epsilon)f|S|}{2\Gamma} \geq \frac{\epsilon\gamma^2 m}{2\Gamma}.$$

Furthermore, if a transaction tx is included in any block computed by an honest party for the first $\ell + 2\Delta$ rounds, by Lemma 25, the chain \mathcal{C} of any honest party contains tx in a block B . Thus, the honest parties will have

$$\ell + 4\Delta + \frac{\epsilon\gamma^2 m}{(1 - 2\epsilon)f} \leq \frac{\epsilon m}{2(1 + \delta)\Gamma^2 f} + \frac{\epsilon\gamma^2 m}{(1 - 2\epsilon)f} \leq (\gamma^2 + 1) \cdot \frac{\epsilon m}{f}$$

as the total wait-time. □