2019 IEEE Symposium on Security and Privacy

Lay Down the Common Metrics: Evaluating Proof-of-Work Consensus Protocols' Security

Ren Zhang
Nervos and *imec-COSIC, KU Leuven*
ren@nervos.org

Bart Preneel
imec-COSIC, KU Leuven
bart.preneel@esat.kuleuven.be

***Abstract*—Following Bitcoin's Nakamoto Consensus protocol (NC), hundreds of cryptocurrencies utilize proofs of work (PoW) to maintain their ledgers. However, research shows that NC fails to achieve perfect chain quality, allowing malicious miners to alter the public ledger in order to launch several attacks, i.e., selfish mining, double-spending and feather-forking. Some later designs, represented by Ethereum, Bitcoin-NG, DECOR+, Byzcoin and Publish or Perish, aim to solve the problem by raising the chain quality; other designs, represented by Fruitchains, DECOR+ and Subchains, claim to successfully defend against the attacks in the absence of perfect chain quality. As their effectiveness remains self-claimed, the community is divided on whether a secure PoW protocol is possible. In order to resolve this ambiguity and to lay down the foundation of a common body of knowledge, this paper introduces a multi-metric evaluation framework to quantitatively analyze PoW protocols' chain quality and attack resistance. Subsequently we use this framework to evaluate the security of these improved designs through Markov decision processes. We conclude that to date, no PoW protocol achieves ideal chain quality or is resistant against all three attacks. We attribute existing PoW protocols' imperfect chain quality to their unrealistic security assumptions, and their unsatisfactory attack resistance to a dilemma between "rewarding the bad" and "punishing the good". Moreover, our analysis reveals various new protocol-specific attack strategies. Based on our analysis, we propose future directions toward more secure PoW protocols and indicate several common pitfalls in PoW security analyses.**

***Index Terms*—blockchain, proof-of-work consensus, incentive compatibility, double-spending, censorship resistance**

I. Introduction

By November 2018, more than six hundred digital currencies leverage *proofs of work (PoW)*, i.e., moderately hard computational tasks, to maintain consensus on a public ledger of transactions [1]. All PoW consensus protocols originate from Bitcoin's *Nakamoto Consensus (NC)* [2], in which participants, called *miners*, compete in generating the latest *block*—a group of new transactions bound with a solution to a computational puzzle. The protocol helps participants reach agreement on a sequence of blocks named the *blockchain*. The miner of each blockchain block is entitled to a *block reward* of new bitcoins to incentivize protocol participation. Remarkably, NC is the first scheme that promises an inalterable public ledger without prior knowledge on participants' identities. Unfortunately, the security of NC is challenged by several studies [3]–[7], in which researchers identify a wide range of strategies that allow attackers with less than 50% of total computing power to rewrite part of the blockchain with high success rate.

Given NC's security weakness, a considerable amount of non-NC PoW protocols [6]–[23] have emerged in the past few years, which all claim to achieve stronger security properties. Nevertheless, in the absence of a systematic evaluation, such advancements remain self-claimed and not widely acknowledged. Moreover, some protocols introduce new issues like lowering the chain-growth rate [24], [25] or facilitating an attacker to create disagreements among the compliant miners [26]. This inconclusive situation also feeds the pessimistic atmosphere surrounding PoW, leading new digital currencies to abandon PoW and turn to other consensus mechanisms such as proofs of stake (PoS), which all rely on stronger security assumptions, yet open new attack vectors [27]–[29].

In this paper, we address this situation and explore the (im)possibility of more secure PoW protocols. Our work and contributions include:

A quantitative security evaluation framework. We identify that NC's key weakness lies in its low *chain quality*, defined as the fraction of blockchain blocks mined by the compliant miners. The unsatisfactory chain quality allows attackers to substitute other miners' blocks from the blockchain with the attackers', which impairs NC's inalterability promise and could be utilized by attackers to cause three kinds of damage: they can (1) gain relative block rewards larger than their fair share with a *selfish mining attack* [6]; (2) spend the same coin more than once with a *double-spending attack*; and (3) force rational miners to collectively censor certain target transactions with a *feather-forking attack* [30].

Accordingly, to verify the self-claimed improvements of recent non-NC protocols and to detect the security flaws in PoW designs, we propose a comprehensive evaluation framework including *chain quality* and three attack-resistance metrics of *incentive compatibility*, *subversion gain* and *censorship susceptibility*, corresponding to the aforementioned attacks.

Generalizing MDP-based methods for analyzing PoW protocols. While Markov decision processes are commonly used to explore an actor's utility-maximizing strategies in a stochastic environment, previous MDP-based analyses mostly focus on NC with a rational, i.e., profit-driven, adversary [4], [31], [32]. We generalize their methods on two dimensions. First, by redefining the attacker's utility, we extend the model to include *byzantine adversaries*, whose goals are not limited to their economic gains. This generalization allows our model

DOI 10.1109/SP.2019.00086

TABLE I
SECURITY ANALYSES BY THE PROTOCOL DESIGNERS AND OUR NEW RESULTS.

Group	Protocol	Designers' analysis	Our results
Better-chain-quality	SHTB [12]	None	New protocol-specific attack strategy
Better-chain-quality	UDTB [18], [21]	Analysis against one attack strategy	New protocol-specific attack strategy
Attack-resistant: reward-all	Fruitchains [20]	Formal analysis against selfish mining assuming some parameters are large enough	Vulnerable to selfish mining and double-spending attacks with reasonable parameters
Attack-resistant: punishment	RS [12], [21]	Analysis against one attack strategy	Vulnerable to censorship attack
Attack-resistant: reward-lucky	Subchains [11]	None	Vulnerable to all three attacks

to capture more real-world attack scenarios, such as censorship or chain quality attacks. Second, by introducing new modeling and acceleration techniques, our MDPs can model more complicated systems and support longer block races than previous works, which enables cross-protocol security comparison.

Moreover, our approach opens the possibility of applying artificial intelligence techniques in analyzing protocol security. By properly simplifying the protocol and confining the attackers' reasonable actions, these techniques enable systematic exploration of a protocol's vulnerabilities with a given attacker goal, which helps improve the protocol design iteratively.

Systematic evaluation of non-NC PoW protocols. Based on their self-claimed properties, we divide PoW protocols claiming to improve NC's security into two groups: *better-chain-quality protocols* and *attack-resistant protocols*, differing in whether they accept imperfect chain quality as a given condition. We then use our framework to evaluate the two groups accordingly. Our findings are summarized as follow:

- *No PoW protocol achieves perfect chain quality facing a strong attacker.* We first evaluate the chain quality of two influential better-chain-quality protocols that are previously unverified: smallest-hash tie-breaking (SHTB) [12] and unpredictable deterministic tie-breaking (UDTB) [18], [21]. Joining the results of previous studies [4], [13], [31], we confirm that an attacker with more than a quarter of total mining power can obtain an unfair fraction of blockchain blocks in all better-chain-quality protocols. We attribute the low chain quality to information asymmetry between the attacker and the compliant miners, which is inherent to the unrealistic security assumptions in PoW protocols, including the participants' unawareness of their own network connectivity and the lack of a globally synchronous clock.
- *No attack-resistant protocol is resistant against all three attacks.* Then we evaluate the attack-resistant protocols based on the metrics of incentive compatibility, subversion gain, and censorship susceptibility. We further divide these protocols into three groups based on their technical approaches: *reward-all protocols*, *punishment protocols* and *reward-lucky protocols*. We choose a representative and most influential protocol from each approach for evaluation: Fruitchains [20], a variant of DECOR+ [12], [21] named reward-splitting protocol (RS), and Subchains [11]. Our analysis shows that all three approaches suffer from certain drawbacks: reward-all protocols remove the attacker's risk of losing block rewards in double-spending attacks; punishment protocols aid feather-forking attacks; reward-lucky protocols facilitate all three attacks. We attribute these empirical results to a dilemma between "rewarding the bad" and "punishing the good".

Our findings show that no better-chain-quality protocol outperforms NC's chain quality in all attacker settings, neither does any attack-resistant protocol outperforms NC in defending against all three attacks. Starting from our identified cruxes hindering substantial improvement in both chain quality and attack resistance, we point out several directions of future improvements towards more secure PoW protocols.

Exposing limitations in existing PoW protocols' security analyses. The unsatisfactory security of PoW protocols roots in the designers' lack of [7], [8], [11], [12], [17]–[19], or incomplete security analyses. Existing analyses are limited either to only one attack strategy [6], [9], [21]–[23], turning its back on the protocol-specific attack strategies, or to one or two security properties [10], [13]–[16], [20], [33], leaving the protocols more vulnerable against other attacker incentives. In addition, our analysis reveals that, in some designers' analysis, certain parameters are artificially anchored to an unrealistic range in order to prove the properties of the protocol, leaving the real-world security unexplored. Of the five protocols we model in this paper, a comparison between our results and the designers' own analyses is summarized in Table I. Our results highlight that PoW protocols' security is not a unidimensional index, but rather a multi-metric property subjects to *the law of the minimum*—security is decided by the weakest point in the design. Therefore, future protocol analyses need to consider a broad strategy space covering the all reasonable actions with a given attacker goal, and incorporate multiple attacks with real-world parameters.

II. NAKAMOTO CONSENSUS'S SECURITY ISSUES AND ALTERNATIVE POW PROTOCOLS

A. Nakamoto Consensus

NC helps all network participants agree on and order the set of confirmed transactions in a decentralized, pseudonymous way. Each block contains its *height*—distance from the hard-coded *genesis block*, the hash value of the *parent block*, a set of transactions, and a nonce. Embedding the parent hash ensures that a miner chooses which chain to mine on before starting to mine. To construct a valid block, miners work on finding the right nonce so that the block hash is smaller than

the *block difficulty target*. This target is adjusted every 2016 blockchain blocks so that on average one block is appended to the blockchain in ten minutes. Compliant miners publish blocks to the network the moment they are found. Miners are incentivized by two kinds of rewards. First, a *block reward* is allocated to the miner of every blockchain block. Second, the value difference between the inputs and the outputs in a transaction is called the *transaction fee*, which goes to the miner who includes the transaction in the blockchain.

When more than one block extends the same preceding block, a miner adopts and mines on the *main chain* that is most computationally challenging to produce, which is commonly, although inaccurately, referred to as the *longest chain*. When several chains are of the same "length", miners choose the first chain they receive. We refer to this *forked* situation where miners work on different parent blocks as a *block race*, an equal-length block race as a *tie*, and blocks of the same height as *competing blocks*. Mining on the longest chain or the first-received block during a tie is denoted as the *compliant strategy* [5], [26], [34], [35]. Blocks that are not on the longest chain are orphaned and discarded by all miners. By convention, Bitcoin users will not consider a transfer of funds settled until it is confirmed by six blocks, including the block containing the transaction. We refer to Narayanan et al. [36] for a complete view of the system.

B. Nakamoto Consensus's Security Issues

Bitcoin's designer believes that the protocol achieves *perfect chain quality*, i.e., as long as more than half of total mining power is compliant, any attempt to substitute blocks from the blockchain fails with large probability [2]. Unfortunately, this belief is disproved by several later studies [3]–[7], which discover a family of strategies to replace the compliant miners' blocks with the attackers' at the end of the blockchain with high success rate. The imperfect chain quality can be directly exploited to manipulate vote results in some blockchains [37].

Moreover, the imperfect chain quality enables a variety of other attacks, differing in the attackers' goals:

- *Selfish mining.* In this attack first analyzed by Eyal and Sirer [6], a *selfish miner* keeps discovered blocks secret and mines on top of them, hoping to gain a larger lead on the public chain of *honest blocks* mined by the compliant miners. The selfish miner publishes the secret chain if it has one block and the public chain catches up, or it has more than one block and the lead is reduced to one. Though risking the reward of the first secret block, once the selfish chain is two blocks ahead of its competitor, the selfish miner can securely invalidate compliant miners' competing blocks. This strategy has been generalized by Sapirshtein et al. [4] and Nayak et al. [5] to a family of strategies.

  This attack allows the selfish miner to gain unfair block rewards. As the attacker's revenue rises superlinearly with the mining power share, rational miners are incentivized to attack collectively for a higher input-output ratio. This situation not only damages the system's decentralized structure, but also raises the success rates of various other attacks.
- *Double-spending.* A successful double-spending attack reverses the payment after the service or goods are delivered. The transaction to the merchant is replaced by a *conflicting transaction* transferring the fund back to the attacker. Double-spending were once believed to be difficult with less than 50% of total mining power [2]. However, a 2016 paper by Sompolinsky and Zohar [32] indicates that an attacker with arbitrarily low mining power can profitably implement the attack by combining it with selfish mining: the attacker mines in secret to perform double-spending attacks, and when there is little hope to orphan six blocks in a row, the attacker publishes the secret blocks to claim the block rewards, switching to selfish mining instead.
- *Feather-forking.* In this attack proposed by Miller [30], the attacker publicly promises to fork the blockchain to invalidate all blocks confirming the target transactions. The attacker will keep mining on the forked chain until the main chain is k blocks ahead. Although the attack is not profitable and the success rate is low with minority mining power, the rational choice for other miners is to join the attacker on the censorship in order to avoid the potential loss.

  A successful attacker can approve and decline transactions at will, becoming the system's de facto owner, which violates the motivation of the permissionless design.

Researchers identify several other attacks against NC [35], [38]–[42]. Nevertheless, these attacks either do not have their roots, as well as their solutions in the consensus protocol, or do not bring realistic threats in the coming decades.

C. Alternative PoW Protocols

A substantial number of alternative PoW protocols have been proposed to address NC's security issues. In this part we split these designs into two groups, better-chain-quality and attack-resistant protocols, based on their claims, and selectively introduce some most influential designs. These two groups are not mutually exclusive. Although we omit non-security-related innovations and hybrid protocols, i.e., protocols that combine PoW with other consensus mechanisms [43]–[45], our security analysis is still applicable to their underlying PoW protocols. We refer interested readers to the recent SoK paper of Bano et al. [28] for a more complete overview of consensus protocols.

1) Better-chain-quality protocols: These designs usually modify NC's *fork-resolving policy*, hoping to reduce the probability that the compliant miners work on the attacker's chain during a block race. The first three designs abandon NC's first-received tie-breaking rule, yet still follow the longest-chain rule, whereas the others abandon both rules.

a) Uniform tie-breaking: Eyal and Sirer suggest during a tie, miners choose which chain to mine on uniformly at random regardless of which one they receive first [6]. This policy is adopted by the PoW component of Ethereum, the cryptocurrency with the second largest market capitalization [46]. Bitcoin-NG, a high-throughput blockchain protocol [47] implemented in two cryptocurrencies Waves [48] and Aeternity [49], also follows uniform tie-breaking policy.

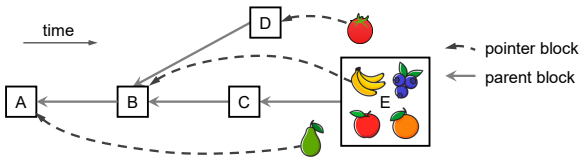

Fig. 1. A Fruitchains execution. Banana's gap is height(E) − height(B) = 2. Tomato is not a valid fruit because its pointer block (D) is orphaned. When $T_o = 3$, pear is not valid even if it is included in E, as its gap reaches T_o.

b) Largest-fee and smallest-hash tie-breaking: Lerner proposes DECOR+, in which during a tie, miners choose the chain whose *tip*, i.e., the last block, has the largest transaction fees, and when multiple tips have the same amount of fees, choose the one with the smallest hash [12]. A variant of DECOR+ is implemented in Rootstock [17], a Bitcoin sidechain [50]. The author believes a *deterministic tie-breaking policy* helps the compliant miners choose the same chain in a tie, thus limiting the attacker's ability.

c) Unpredictable deterministic tie-breaking: In Byzcoin [18], Kokoris-Kogias et al. recommend that ties are resolved deterministically via a pseudorandom function taking all competing blocks as inputs. This tie-breaking policy is also described by Camacho and Lerner in an updated version of DECOR+ [21]. Within this policy, the attacker can neither determine whether a secretly-mined block can win a tie with unfair possibility before all competing blocks are mined, nor split the compliant mining power.

d) Publish or perish: Zhang and Preneel present a design *Publish or Perish*, in which forks are resolved by comparing all chains' *weights* [13]. Blocks published after their competitors do not contribute to the weight of its chain, and blocks that incorporate links to their parents' competitors are appreciated more. Consequently, a block that is kept secret until a competing block is published contributes to neither or both branches, hence it confers no advantage in winning the block race.

e) Others: Other better-chain-quality protocols include the GHOST protocol designed by Sompolinsky and Zohar [33] and Chainweb by Martino et al. [23].

2) Attack-resistant protocols: These designs usually modify NC's *blockchain topology* and *reward distribution policy*, hoping to reduce the attacker's profitability or to reduce the compliant miners' losses. They can be categorized into three types: the first two types issue rewards based on the block's topological position in the blockchain, whereas the third type issues rewards based on the block content.

a) Reward-all protocols: In these designs, most of recent PoW solutions receive a fraction of a full reward, although some of them may not contribute to the transaction confirmation. Consequently, the compliant miners' losses due to malicious orphaning of their blocks are compensated.

Fruitchains by Pass and Shi [20] distributes rewards to all recent *fruits*, which are parallel products of block mining. Similar to "a block candidate is a block if its hash's first l bits are smaller than a predefined target", the candidate is a *fruit* if its hash's *last* l bits are smaller than another target. Although generated from the same mining process, fruits and blocks have different functionalities. Each block embeds an ordered fruit list, similar to each block in NC embeds an ordered transaction list; transactions are embedded in fruits instead. Transactions are ordered based on their first fruit appearances in the blockchain. In addition to the transactions, each fruit contains a *pointer* to a recent main chain block which the fruit miner is certain will not be orphaned. A fruit is valid if its pointer block is not orphaned, or its *gap*—the height difference between its pointer block and the main chain block contains the fruit—is smaller than a timeout threshold T_o. All valid fruits receive the same reward and blocks receive nothing. This incentive mechanism is also adopted by Thunderella, a blockchain design of the same authors [43].

Other designs of this type include the PoW component of Ethereum, the Inclusive protocol by Lewenberg et al. [10], SPECTRE by Sompolinsky et al. [14], Meshcash by Bentov et al. [15], and PHANTOM by Sompolinsky and Zohar [16].

b) Punishment protocols: As it is often hard to tell which of the competing blocks are mined by the attacker, these designs forfeit rewards of all competing blocks to deter attacks. In DECOR+, the block reward is split evenly among all competing blocks of the same height [12], [21]. The authors propose some other punishment rules for suspected malicious behaviors. Bahack suggests another punishment protocol [7].

c) Reward-lucky protocols: These designs selectively reward PoW solutions, hoping that these solutions serve as anchor points to stabilize the blockchain. Subchains by Rizun demands miners to broadcast *weak blocks*, i.e., block candidates with larger difficulty target, in addition to blocks [11]. Weak blocks also count in chain length and contribute to the transaction confirmation, though receive no reward. Subchains follows NC's longest chain and first-received rule. Bobtail by Bissias and Levine [22] is another reward-lucky protocol.

III. Evaluation Framework and Security Model

As non-NC PoW protocols' security improvements remain self-claimed, we propose our evaluation framework in order to investigate whether they have fixed NC's weaknesses, and to shed light on the possibility of such improvement.

A. Evaluation Framework

We present four metrics for a more comprehensive view on PoW protocols' security. This is not an exhaustive list of all metrics proposed in the literature, but rather a comparative framework with NC as the benchmark. In particular, though the chain-growth and the common-prefix properties are also used to quantify consensus protocol security [3], [15], [25], [51], they are not included, because the attack vectors on these properties are only introduced by certain non-NC protocols.

1) Chain quality: This metric measures the difficulty to substitute the honest main chain blocks. In line with previous research [3], [25], [51], we define the chain quality Q as the expected lower bound on the fraction of honest main chain blocks, given that the attacker controls a fraction of total mining power α. Defining B_c and B_a as the total number

of main chain blocks mined by the compliant miners and the attacker respectively, and s the attacker's strategy, we have:

$$Q(\alpha) = \min_{s} \lim_{t\to\infty} \frac{B_c}{B_a + B_c} \ .$$

Ideally, $Q(\alpha) = 1 - \alpha$, namely the attacker gets main chain blocks at most proportional to the mining power. A protocol's chain quality is not related to its reward distribution policy.

2) Incentive compatibility: This metric measures a protocol's selfish mining resistance. It is defined as the expected lower bound on the *relative revenue* of the compliant miners [4]–[7], [13], [26], [31], namely:

$$I(\alpha) = \min_{s} \lim_{t\to\infty} \frac{\sum R_c}{\sum R_a + \sum R_c} \ ,$$

where $\sum R_a$ and $\sum R_c$ are the cumulative rewards received by the attacker and the compliant miners, respectively. Incentive compatibility shares the same ideal value $1 - \alpha$ with chain quality. Unlike chain quality, all three attack resistance metrics are tightly related to the reward distribution policy.

3) Subversion gain: This metric measures the profitability of double-spending attacks, which is quantified as the time-averaged illegal upper bound profit in a specific attack model, in line with several previous papers [26], [31], [32]. In this model, every honest block contains a *payment transaction* to the merchant, whose conflicting version is embedded in the block's secret competitor, if the competitor exists. The service or goods are delivered when the block containing the payment transaction reaches σ confirmations, with $\sigma = 6$ in Bitcoin, or the attacker gives up on attacking this block. In the former case, if the payment transaction is later invalidated, for every block that is orphaned after confirmation, the attacker receives a double-spending reward V_{ds}, in the unit of block rewards. In other words, if the attacker successfully orphans k blocks in a row, the double-spending reward is defined as

$$R_{\mathrm{ds}}(k, \sigma, V_{\mathrm{ds}}) = \begin{cases} 0, & k < \sigma \\ (k + 1 - \sigma)V_{\mathrm{ds}}, & k \geq \sigma \end{cases} \ , \quad (1)$$

where $k + 1 - \sigma$ is the number of σ-confirmation blocks that are orphaned. In addition, if the first payment transaction is invalidated before reaching σ confirmations, $R_{\mathrm{ds}} = 0$. The attacker receives no punishment for failed double-spending attempts, because if an attack fails, the service or goods will be delivered eventually, compensating the attacker's loss. This metric captures multiple aspects of a protocol's double-spending resistance. First, incorporating $\sum R_a$ forces the attacker to balance the risk of losing block rewards with the double-spending gain. Second, the merchant is allowed to delay delivery if the conflicting transaction is broadcast before σ confirmations, counteracting the attack. Third, longer forks, which cause more damage in reality, result in higher rewards.

The subversion gain of the attacker is defined as:

$$S(\alpha, \sigma, V_{\mathrm{ds}}) = \max_{s} \lim_{t\to\infty} \frac{\sum R_a + \sum R_{\mathrm{ds}}}{t} - \alpha,$$

where t represents the lasting time, measured as the number of block generation intervals; α is the time-averaged mining reward without the double-spending attack. Ideally, the attacker complies with the protocol to avoid losing any block reward, namely $S(\alpha, \sigma, V_{\mathrm{ds}}) = 0$. However, an attacker is always incentivized to deviate as long as V_{ds} is large enough.

4) Censorship susceptibility: Inspired by feather-forking attacks, we measure censorship susceptibility as the maximum fraction of income loss the attacker incur on compliant miners in a censorship retaliation attack. We choose not to incorporate the attacker's economic loss, as the retaliation does not happen if the censorship threat succeeds. As long as the other miners are convinced of the attacker's determination, the only factor affecting their strategy is the expected loss of not cooperating. Unlike feather-forking, in which the retaliation starts after receiving the block containing the target transaction, in our model, the attack is initiated as soon as compliant miners start mining the block. This setting is practical, as the attacker can learn the transaction inclusion as soon as the mining starts by eavesdropping in compliant mining pools. Another difference with feather-forking is that we remove the reliance on the parameter k by allowing the attacker to drop the falling-behind chain and try to orphan the next honest block at any time. As the attacker's goal is to maximize the compliant miners' loss, mining on a falling-behind chain is not always optimal. Our generalized setting captures multiple attack scenarios. For example, in an extreme form of the attack, attackers degrade the system's availability by replacing honest blocks with empty blocks, delaying all transactions' confirmation.

A protocol's censorship susceptibility is defined as:

$$C(\alpha) = \max_{s} \lim_{t\to\infty} \frac{\sum O_c}{\sum O_c + \sum R_c} \ ,$$

where $\sum O_c$ is the compliant miners' cumulative reward loss due to the attack, in the unit of block rewards. Ideally, $C(\alpha) = 0$, namely the compliant miners have no risk rejecting a censorship request.

B. Threat Model

We follow the threat model of most studies on PoW security [3]–[7], [9], [13], [26], [31], [32], [52]. In this model, there is only one colluding pool of malicious miners, denoted as "the attacker", with less than half of total mining power. All other miners are compliant. This is the strongest form of the attacks as multiple attackers cause more damage when combining their mining power. We do not consider the effect of transaction fees as in [35], [42], [53]. Neither do we incorporate the difficulty adjustment mechanism as in [39].

In terms of network connectivity, the attacker cannot drop other miners' messages or downgrade their propagation speed. However, the attacker may, after seeing a compliant miner's message, send a new one to certain miners that arrives before the original message. The propagation delay is modeled as a fixed natural orphan rate, as in [31]. Unfortunately, as many protocols we evaluate are under development and their parameters are not specified, it is difficult to estimate their orphan rates. Therefore we assume all protocols in this work have the same expected block interval and zero natural orphan rate, in order to ensure a fair comparison on the protocol level.

In this model, the following result has been proven [4], [52]: if the protocol follows the longest-chain rule and the attacker is rational, there are at most two active chains at any given time: a *public chain*, and at most one *attacker chain*, whose last several blocks might be hidden from the compliant miners. Any more-than-two-chain strategy decreases the attacker's effective mining power, therefore is strictly dominated by a two-chain strategy. We refer to the last common block of these two chains, namely the last block recognized by all miners, as the *consensus block*.

C. Modeling Mining Processes as MDPs

An MDP is a discrete time stochastic control process that models the decision making in situations where outcomes are partly random and partly under the control of a decision maker. To model a system as an MDP, we need to encode all status and history information that might influence the strategic player's decisions into a *state*, and the player's available decisions into several *actions*. Moreover, a *state transition matrix* describes the probability distribution of the next state over every $(state, action)$ pair. At last, when certain $(state, action, new\_state)$ transition happens, a *reward* is allocated to the player to facilitate utility computation.

In line with previous studies [4], [13], [26], [31], [32], mining is modeled as a sequence of steps. The MDP *state* describes the blockchain's status at the beginning of a step, which incorporates all information that might affect the attacker and the compliant miners' decisions, e.g., the lengths of competing chains, the miners of the last several blocks, and the number of unpublished attacker blocks. Encoding a blockchain status into a state is challenging, as despite the sparseness of the transition matrices and our optimization, an MDP solver gives the exact solution only when the number of states is less than about 10^7. In each step, the attacker first decides how many secret blocks to publish. Next, the rewards are distributed for certain blocks if all miners agree that these blocks are settled, either as main chain blocks, orphans or *uncles*, i.e., orphans that are referred to in the main chain. Afterwards, all miners start mining. The compliant miners choose which chain to mine on based on public information, whereas the attacker may choose either chain. The *action* in an MDP describes the attacker's choices on how many blocks to publish and which chain to mine on. A new block is then mined by either the attacker or the compliant miners, with probability distribution according to their mining power shares. New honest blocks are published immediately, whereas the attacker decides whether to publish his new block at the beginning of the next step. The attacker's old blocks published in the next step might reach the compliant miners before the new honest block. The MDP *state transition* is triggered by the new mining event. The rationale behind this publish-reward-mine-found sequence is that rational decisions may only change when a new block is available [4], [7]. Whenever it is infeasible to model the exact system, we choose to favor the compliant miners and limit the attacker's ability, ensuring the attacker's utility is achievable in reality to better demonstrate the protocols' weaknesses.

IV. Chain Quality Analysis on Better-Chain-Quality Protocols

This section evaluates the chain quality $Q(\alpha)$ of NC, uniform tie-breaking (UTB), smallest-hash tie-breaking (SHTB), unpredictable deterministic tie-breaking (UDTB) and Publish or Perish (PoP). We do not consider largest-fee tie-breaking, as it enables a malicious miner to locally generate a huge-fee transaction and to embed it in the miner's own block to increase the chance of winning a tie. Neither do we consider GHOST, as it behaves identically to NC when the network delay is negligible [33]. At last, we leave the evaluation of DAG-based protocols, such as [23], to future work as the notion of chain quality is not directly applicable to them.

When orphaned blocks receive no reward and the main chain blocks receive full rewards, the chain quality is equivalent to the compliant miners' relative revenue. Therefore, we implement this reward distribution policy in all MDPs of this section, and define the utility as the attacker's relative revenue $1 - Q(\alpha)$, in order to find the chain quality.

This equivalence also allows us to reuse the relative revenue MDP designs of previous studies. We re-implement the NC, UTB and PoP MDPs as described in [4] and [13]. Our implementation can model block races longer than previous studies, as we accelerate the programs by allocating memory only once before assigning values to the state transition matrices. In this section, we first model the mining process of SHTB and UDTB, and then present the evaluation results.

A. Modeling SHTB

The key challenge of modeling SHTB is to encode in a state the hashes of the latest blocks, as compliant miners resolve ties via comparing these hashes. Unfortunately, a block hash is usually a 256-bit value; encoding which makes the total number of states too large to be solvable. Therefore, we split the hash value space into a small number of *regions* and only encode the hash region number. When comparing two hashes from the same region, we consider the public chain tip to be smaller, which favors the compliant miners. As this simplification discourages the attacker, our MDP computes an upper bound on SHTB's chain quality. We defer the detailed MDP design to Appendix A.

B. Modeling UDTB

The main challenge is to model the pseudorandom function (PRF) determining a tie's winner. We address this challenge by introducing a binary field *tie* in the state representation, denoting whether the public chain tip has priority over its competitor after applying the PRF. This field is meaningful when the attacker chain is no shorter than the public chain. Every time the public chain tip is updated, it has equal probability to be 0 or 1. The design can be found in Appendix B.

C. Evaluation Results

1) Solving for the optimal policies: Our MDPs output the attacker's optimal policy and the expected fraction of main chain blocks following this policy, namely $1 - Q(\alpha)$, allowing

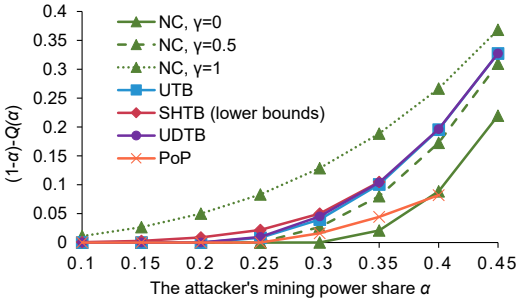

Fig. 2. The difference between the chain quality $Q(\alpha)$ and the ideal value $1-\alpha$ of NC, UTB, SHTB, UDTB and PoP. Larger number indicates worse performance. $Q(\alpha)$ does not converge for PoP and SHTB when $\alpha = 0.45$ and $\alpha \geq 0.4$, respectively.

us to compute $Q(\alpha)$. Besides α, another input in NC is γ, defined as the proportion of compliant mining power that works on the attacker chain during a tie. We compute $Q(\alpha)$ for all five protocols with α between 0.1 and 0.45 with interval 0.05. Three different γ values are chosen for NC: 0, 0.5, and 1. The fail-safe parameter k in PoP is set to 3, following the authors' recommendation [13].

For NC, UTB and UDTB, we set the maximum block race length, denoted as $l_{\max}$, to 160, which is large enough so that $Q(\alpha)$'s lower and upper bounds differ in less than 4×10^{-5} for all inputs. The detailed computation of these bounds can be found in Sect. 4.2 of [4]. For PoP, $l_{\max}$ is set to 30, which is larger than the value 12 in the authors' implementation [26]. For SHTB, we set $l_{\max}$ to be 40 and split the valid hash space into 15 equal-size regions. Once $l_{\max}$ is reached, the attacker is forced to publish the attacker chain and end the block race. For the latter two protocols, we check the convergence by examining whether the results are affected if $l_{\max}$ decreases by two. Data points that do not converge are discarded.

2) Chain quality: Our results of NC, UTB and PoP in Fig. 2 match those from previous studies [4], [13], [31]. We list our new insights as follows.

Result 1: UTB and UDTB's $Q(\alpha)$ are almost identical; they perform no better than NC when $\gamma \leq 0.5$ for all inputs.

For all our inputs, UTB's and UDTB's $Q(\alpha)$ differ in at most 1%. UDTB may outperform UTB when natural forks happen frequently, as these forks are resolved faster in UDTB due to the compliant miners' convergence. UTB's and UDTB's unsatisfactory performance is attributed to the following protocol-specific strategy: as neither policy takes the block receiving time into consideration, an attacker who keeps mining from behind the public chain may still win the block race with a tie. Consequently, their chain quality is lower than that of NC when $\gamma = 0.5$.

Result 2: SHTB achieves the lowest chain quality among all better-chain-quality protocols.

An examination of the optimal strategies reveals the cause of SHTB's poor chain quality. In SHTB, when $\alpha = 0.1$, the optimal action when "the attacker finds a smallest-hash-region block before the compliant miners find anything" is to keep mining privately, whereas in all other protocols except NC, $\gamma = 1$, the weak attacker publishes the block. In other words, resolving ties by comparing hashes allows the attacker to better estimate the probability of winning, hence he is more inclined to deviate when the odds are in favor. Moreover, SHTB enables "catching up from behind" strategy like UTB and UDTB.

TABLE II
THE PROFITABLE THRESHOLD *PT* OF NC, UTB, SHTB, UDTB AND POP.

Protocol	*PT*	Protocol	*PT*
NC, $\gamma = 0$	0.3333	SHTB (upper bounds)	0.0652
NC, $\gamma = 0.5$	0.2500	UDTB	0.2321
NC, $\gamma = 1$	0.0000	PoP	0.2500
UTB	0.2321		

3) Profitable threshold: We calculate the *profitable threshold (PT)*, the maximum α that achieves the ideal chain quality $1-\alpha$ and display the results in Table II.

Result 3: To date, no PoW protocol achieves the ideal chain quality when $\alpha > 0.25$.

SHTB's actual PT should be zero, because as long as a secret block's hash is small enough, the probability of winning a tie can be arbitrarily high, encouraging the attacker to withhold the block. The seemingly above-zero result is because we are unable to encode the hash to arbitrary granularity.

Result 4: No protocol modification outperforms NC, $\gamma = 0$ when $\alpha \leq 0.39$.

NC, $\gamma = 0$ achieves the best chain quality for all $\alpha \leq 0.35$ in Fig. 2. It is only outperformed by PoP when $\gamma \geq 0.4$. We locate the exact value where PoP starts to outperform NC with a binary search: in both PoP and NC, $Q(0.3901) = 0.5372$.

D. What Goes Wrong: Information Asymmetry

We attribute NC's poor chain quality to the protocol's incapability in distinguishing the honest chain from the attacker chain, due to information asymmetry. When two competing chains simultaneously emerge, no information can help the compliant miners identify the attacker chain, or even whether there is an attacker chain, as the fork might be caused by a temporary network partition. In contract, possessing information of both chains, the attacker makes more informed decisions of "gambling" only when the odds are in favor. Since this information asymmetry is not addressed in non-NC protocols, their attempts to raise the chain quality remain unsatisfactory.

Unfortunately, we believe it is difficult to solve this information asymmetry within PoW protocols' security assumptions. In these assumptions, compliant miners can only rely, almost exclusively, on limited public information, namely the blockchain topology and block content, to choose which chain to mine on. While other public information, such as the network partition status, which is highly likely to be available to all miners in reality, as well as the compliant miners' private information such as their network connectivity or the difference between a block's timestamp and its receiving time, is ignored in identifying the attacker chain. The attacker, on the other hand, is able to act on all available information. In other

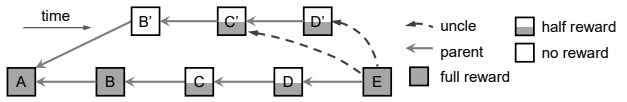

Fig. 3. An RS execution. $\text{gap}(\text{C}') = \text{height}(\text{E}) - \text{height}(\text{C}') = 2$. When $T_o = 3$, B' is not visible even if it is referred to in E as its gap reaches T_o.

words, the information asymmetry is anchored and intensified in these protocols through their unrealistic and inconsistent security assumptions.

V. Incentive Compatibility Analysis on Typical Attack-Resistant Protocols

In the following sections, we analyze the attack resistance of NC and three most influential designs, one from each type of attack-resistant protocols introduced in Sect. II-C2. For reward-all and reward-lucky protocols, we choose Fruitchains and Subchains, respectively. For punishment protocols, we implement our own variant of DECOR+ named *reward-splitting protocol (RS)*. Unlike DECOR+, RS follows NC's longest chain and first-received fork-resolving policy. This modification excludes the influence of the chain quality from our attack resistance analysis, as all four protocols in comparison share the same chain quality. Most insights we gain are direct generalizable to all protocols of the same type.

A. Modeling Fruitchains

We use $\textit{Ratio}_{\text{f2b}}$ to denote the ratio of fruit difficulty target to block difficulty target. For example, $\textit{Ratio}_{\text{f2b}} = 2$ means that of all the *units*—mining products, two thirds are fruits and one third are blocks in expectation.

The main challenge of modeling Fruitchains is to encode each fruit's pointer block. The number of states grows exponentially with the number of steps if we encode all possible choices of each fruit. To address this complexity, we assume all compliant miners know when the block race starts and act optimally to avoid honest fruits being orphaned. Moreover, the attacker's action to cause a tie is disabled so that no honest fruit points at attacker-chain blocks. These assumptions are in favor of the compliant miners. Consequently, incentive compatibility is computed as an upper bound, while subversion gain and censorship susceptibility are computed as lower bounds. Our Fruitchains MDP design can be found in Appendix C.

B. Defining and Modeling RS

In RS, we define a block's *gap* as the height difference between the first main chain block that refers to the block and the block itself. A main chain block's gap is defined as zero. This definition, unlike that of Fruitchains, enables an accurate modeling of our protocol. A block is *visible* if its gap is strictly smaller than the *timeout threshold* T_o. Each block reward is split among all visible blocks of the same height. Other reward-forfeiting mechanisms of DECOR+ are omitted as they are related to its own fork-resolving policy. Therefore, RS's numerical results are not the same as those of DECOR+.

To model RS, we observe that when the attacker wins a block race, it is uncertain whether the orphaned honest blocks are rendered invisible, as they might still be included in the blockchain as uncles. Therefore, we introduce an extra field *history*, a string of at most $T_o - 1$ bits, in our state representation to encode blocks whose rewards are not settled prior to the current block race. Each bit in *history* denotes the blockchain's status at a specific height. Interested readers can find the MDP design in Appendix D.

C. Modeling Subchains

The ratio of weak block difficulty target to block difficulty target is denoted as $\textit{Ratio}_{\text{w2b}}$. Note that $\textit{Ratio}_{\text{w2b}}$ is not equivalent to $\textit{Ratio}_{\text{f2b}}$ in Fruitchains. In Fruitchains, a unit is a fruit as long as the fruit target is met; in Subchains, a unit is a weak block when the weak-block target is met and the block target is *not* met. When $\textit{Ratio}_{\text{w2b}} = 2$, half of the units are weak blocks while the other half are blocks in expectation.

A straightforward encoding of a Subchains state includes both chains' block/weak-block mining sequences, in which the number of states grows exponentially with the block race length. To compress the state space, we observe that in all outcomes of a block race, the public chain is either adopted or abandoned by both miners as a whole. Similar argument applies to the public chain's competing attacker-chain units. Therefore, we encode only the number of blocks in both chains, the attacker chain's last three units and the length difference between the two chains instead of two full mining sequences. This simplification limits the attacker's ability: the attacker can keep no more than three private units after every publication. Hence our Subchains MDP favors the compliant miners. The complete MDP design is in Appendix E.

D. Evaluation Results

Our MDPs output the attacker's optimal strategies and their expected relative revenue, namely $1 - I(\alpha)$. For all three protocols, we compute $I(\alpha)$ with α between 0.1 and 0.45 with interval 0.05 and $\gamma = 0, 0.5$ and 1, except that our Fruitchains MDP does not support $\gamma = 0.5$.

1) Fruitchains: Fruitchains is evaluated with the following set of parameters. $\textit{Ratio}_{\text{f2b}}$ is set to 1 so that the expected number of fruits equals that of blocks, which is the simplest case. The maximum block race length $l_{\max}$ is set to 20. Two different T_o values, 7 and 13 are selected so that we can verify whether a larger T_o results in a higher $I(\alpha)$. In practice, T_o should be no bigger than $\sigma + 1$, where σ is the confirmation threshold, otherwise an attacker can start mining a competing chain to double-spend a confirmed transaction without risking any fruit rewards. Hence the maximum T_o required by Bitcoin's six-confirmation convention and Ethereum's twelve-confirmation convention are 7 and 13, respectively. Other MDP thresholds are set so the probability that these thresholds are reached before $l_{\max}$ is around one percent. The attacker is forced to publish the entire chain if any threshold is reached. The results can be found in the first four data lines of Table III.

TABLE III

INCENTIVE COMPATIBILITY $I(\alpha)$ OF FRUITCHAINS, COMPUTED AS UPPER BOUNDS, SELECTIVELY SHOWN. ENTRIES THAT PERFORM WORSE THAN NC ARE IN RED ITALIC. $I(0.1) = 0.9$ FOR ALL $(T_o, \mathit{Ratio}_{\mathrm{f2b}}, \gamma)$ COMBINATIONS.

$(T_o, \mathit{Ratio}_{\mathrm{f2b}}, \gamma) \setminus \alpha$	0.15	0.2	0.25	0.3	0.35
(7,1,0)	*0.8494*	*0.7961*	*0.7356*	*0.6614*	*0.5658*
(7,1,1)	0.8493	0.7956	0.7337	0.6557	0.5532
(13,1,0)	0.8500	*0.7997*	*0.7472*	*0.6864*	*0.6068*
(13,1,1)	0.8500	0.7997	0.7470	0.6854	0.6036
(13,2,0)	0.8500	*0.7997*	*0.7472*	*0.6866*	*0.6072*
(13,2,1)	0.8500	0.7997	0.7470	0.6856	0.6040
(13,0.5,0)	0.8500	*0.7997*	*0.7472*	*0.6864*	*0.6065*
(13,0.5,1)	0.8500	0.7997	0.7470	0.6853	0.6033

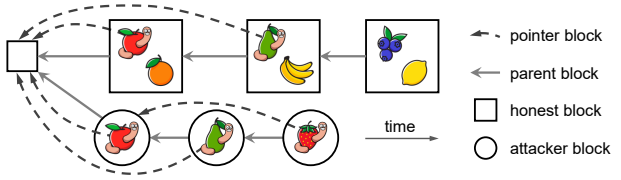

Fig. 4. Selfish mining in Fruitchains, $T_o = 3$. Attacker fruits mined before the T_o-th attacker block are embedded in both chains, whereas honest fruits are only embedded in honest blocks. The attacker loses only the strawberry if losing the block race; however, if the attacker wins the race with $\geq T_o$ attacker blocks, all honest fruits are invalidated.

Result 5: In terms of $I(\alpha)$, Fruitchains performs worse than that of NC for various parameter choices when $\gamma = 0$.

In NC, when $\gamma = 0$, a weak attacker publishes the blocks immediately after they are mined, giving up the temporary lead to avoid losing the block rewards. In contrast, in Fruitchains, as blocks receive no reward, the attacker has no incentive to publish any blocks when neither chain reaches length T_o. This property encourages more audacious block-withholding behaviors aiming to orphan all honest fruits with a long attacker chain. Moreover, this property decreases the profitable threshold to zero: the attacker can withhold blocks as long as the attacker chain is in the lead, regardless of how small α is. An examination of the optimal strategies verifies our inference.

Fruitchains performs better than NC when $\gamma = 1$. This is because in Fruitchains—unlike in NC—winning a block race with a short chain does not increase the attacker's relative revenue.

Result 6: In Fruitchains, $I(\alpha)$ increases along with T_o, at the price of longer transaction confirmation delay.

As T_o increases, the chance that the attacker chain reaches T_o before the public chain decreases, limiting the attacker's unfair relative revenue. According to the authors, $I(\alpha)$ gets arbitrarily close to the ideal value $1-\alpha$ with a large enough T_o. Unfortunately, as $T_o \leq \sigma + 1$, σ must increase along with T_o, resulting in longer transaction confirmation time. Fruitchains's authors have not specified the value of T_o.

Next we study the influence of $\mathit{Ratio}_{\mathrm{f2b}}$ on $I(\alpha)$. Two other $\mathit{Ratio}_{\mathrm{f2b}}$ values, 2 and 0.5, are chosen for $T_o = 13$. The results can be found in the last four lines of Table III.

TABLE IV

$I(\alpha)$ AND PROFITABLE THRESHOLD *PT* OF REWARD-SPLITTING PROTOCOL (RS). ENTRIES PERFORM WORSE THAN NC ARE IN RED ITALIC. OMITTED ENTRIES, INCLUDING ALL ENTRIES WITH $\alpha \leq 0.25$, REALIZE THE IDEAL VALUE $I(\alpha) = 1 - \alpha$.

$(T_o, \gamma) \setminus \alpha$	0.3	0.35	0.4	0.45	*PT*
(3,0)		0.6084	0.4842	*0.3097*	*0.3022*
(3,0.5)		0.5997	0.4534	0.2575	0.3021
(3,1)	0.6921	0.5771	0.4292	0.2406	0.2918
(6,0)			0.5283	0.3454	0.3549
(6,0.5)			0.5056	0.2945	0.3509
(6,1)		0.6397	0.4899	0.2816	0.3428
(9,0)			0.5566	0.3690	0.3752
(9,0.5)			0.5388	0.3210	0.3702
(9,1)			0.5269	0.3098	0.3647

Result 7: In Fruitchains, $I(\alpha)$ increases along with $\mathit{Ratio}_{\mathrm{f2b}}$, at the price of more repeating transactions in different fruits.

This result is similar to that of the Newton-Pepys problem [54]: a higher $\mathit{Ratio}_{\mathrm{f2b}}$ lowers the execution's variance, thus favors the compliant miners with majority mining power. However, the gain comes with a trade-off: more parallel fruits contain more repeating transactions, which demands better network optimization to avoid wasting bandwidth.

2) RS: Three different T_o values are chosen: 3, 6 and 9. $T_o = 6$ here is roughly equivalent to $T_o = 7$ in Fruitchains: in both cases, the first honest unit's reward is removed when the sixth attacker chain block is accepted by all miners. The profitable thresholds are also calculated. We set $l_{\max} = 30$ and all data points converge. The results are shown in Table IV.

Result 8: In RS, $I(\alpha)$ increases along with T_o.

RS with $T_o = 3$ outperforms Fruitchains with $T_o = 7$ for all inputs. $I(\alpha)$ is further improved when T_o increases. For any $\alpha < 0.5$, RS is able to achieve the ideal $I(\alpha)$ with a large enough T_o, rather than getting asymptotically close to the ideal value as in Fruitchains. This is because unlike Fruitchains where block withholding has no risk, in RS half of the secret blocks' rewards are at risk even if the attacker wins the block race. Therefore, when the potential risk outweighs the relative revenue gain in selfish mining, the attacker follows the compliant strategy and $I(\alpha) = 1 - \alpha$.

3) Subchains: The maximum numbers of blocks in both chains are set to 20. The length difference of the chains diff_u is set in range $[-5, 20]$. The attacker is forced to end the block race once the border numbers are reached. Two different $\mathit{Ratio}_{\mathrm{w2b}}$ values, 2 and 3 are selected to verify whether a larger weak-block-to-block ratio results in a higher $I(\alpha)$. The results are selectively displayed in Table V.

Result 9: In Subchains, $PT = 0$ for all parameter combinations. In other words, Subchains is not incentive compatible regardless of how weak the attacker is.

We examine the optimal strategies and discover a series of attacks. For example, when the first several units in a block race are attacker weak blocks, the attacker will not publish them regardless of how small α is, as weak blocks

TABLE V
$I(\alpha)$ OF SUBCHAINS, UPPER BOUNDS, SELECTIVELY SHOWN. ENTRIES PERFORM WORSE THAN NC ARE IN RED ITALIC.

$(Ratio_{\mathrm{w2b}}, \gamma) \setminus \alpha$	0.1	0.15	0.2	0.25	0.3
(2,0)	*0.8990*	*0.8467*	*0.7922*	*0.7342*	*0.6712*
(2,0.5)	*0.8970*	*0.8426*	*0.7853*	*0.7241*	*0.6570*
(2,1)	0.8889	0.8235	0.7500	0.6667	0.5714
(3,0)	*0.8987*	*0.8456*	*0.7895*	*0.7288*	*0.6613*
(3,0.5)	*0.8960*	*0.8401*	*0.7804*	*0.7156*	*0.6432*
(3,1)	0.8889	0.8235	0.7500	0.6667	0.5714

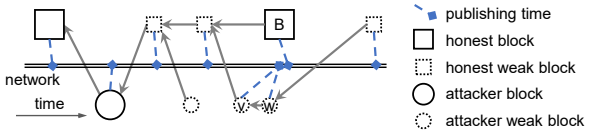

Fig. 5. A typical selfish mining strategy for a weak attacker in Subchains. The attacker withholds only weak blocks to invalidate honest blocks. In this example, honest block B is invalidated by attacker weak blocks v and w.

receive no reward. These weak blocks are used to invalidate honest blocks, thus increasing the attacker's relative revenue. Consequently, Subchains is never incentive compatible.

Subchains always performs worse than NC with $\gamma < 1$. Two protocols are equally bad when $\gamma = 1$, because in both protocols, every attacker unit can orphan an honest unit without any risk.

Result 10: In Subchains, $I(\alpha)$ decreases as $Ratio_{\mathrm{w2b}}$ increases.

Unfortunately, a larger $Ratio_{\mathrm{w2b}}$ does not help $I(\alpha)$. This is because more weak blocks give the attacker more windows to orphan honest blocks with attacker weak blocks.

VI. SUBVERSION GAIN ANALYSIS

A. Modeling Subversion Gain

Similar to previous works [26], [31], [32], all subversion gain MDPs output average reward per step, rather than the relative revenue, as the latter value has no practical meaning.

1) NC and RS: Our NC subversion gain MDP extends previous works [26], [31], [32] by allowing the merchant to delay delivery if the conflicting transaction is broadcast before the first payment transaction in a block race receives σ confirmations. In order to carry this "early publication" information to reward allocation, we introduce an extra field *matched* in the state representation, which is a binary value encoding whether the earliest attacker block in this block race is published to cause a tie before σ confirmations. When all miners accept some attacker blocks into the blockchain and *matched* $=$ false, the attacker receives double-spending rewards R_{ds} in addition to the block rewards, which is defined according to Eqn. (1) in Sect. III-A3. RS's subversion gain MDP follows the same modifications.

2) Fruitchains: Fruitchains's subversion gain MDP issues R_{ds} according to Eqn. (1) when the attacker wins a block race. There is no need to introduce a *matched* field, as our

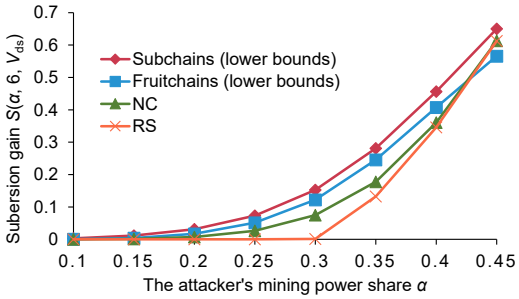

Fig. 6. Subversion gain $S(\alpha, \sigma, V_{\mathrm{ds}})$ of four protocols when $\gamma = 1, \sigma = 6, R_{\mathrm{ds}} = 3, l_{\max} = 24$. Larger number indicates worse performance.

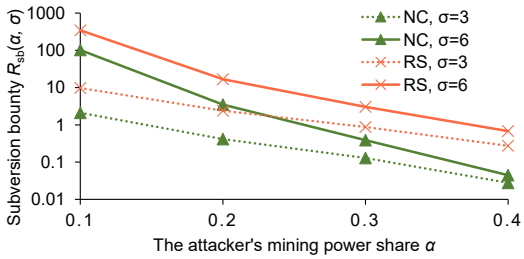

Fig. 7. Subversion bounty $R_{\mathrm{sb}}(\alpha, \sigma)$ of NC and RS, $\gamma = 0.95$.

Fruitchains MDP does not allow publishing part of the attacker chain. Note that k and σ in the equation only count the number of blocks, as fruits do not contribute to the transaction ordering. The outputs are normalized to average reward per confirmation, namely per block, rather than per unit, in line with other protocols in comparison.

3) Subchains: As our Subchains MDP does not encode the public chain's length, we assume the service or goods are delivered when the transaction is confirmed by σ' blocks, so that $\sigma' \times Ratio_{\mathrm{w2b}}$ is roughly equivalent to σ in other protocols. In line with other protocols' "one unit of block reward per confirmation" rule, each main chain block receives $Ratio_{\mathrm{w2b}}$ reward units. The double-spending reward R_{ds} is also multiplied by $Ratio_{\mathrm{w2b}}$ to incorporate the transactions embedded in weak blocks and later reverted. A *matched* field is added to the state encoding, similar to that of NC and RS.

B. Evaluation Results

1) Subversion gain: We display results from one set of parameters and inputs that cover all new insights in Fig. 6. The attacker has the strongest propagation advantage, i.e., $\gamma = 1$. We set $\sigma = 6$ following Bitcoin's convention. R_{ds} is set to 3, which is of the same order of magnitude as the block reward, forcing the attacker to balance two kinds of rewards. The timeout thresholds T_o are set to 7 and 6 in Fruitchains and RS, respectively. $Ratio_{\mathrm{f2b}}$ and $Ratio_{\mathrm{w2b}}$ are set to 1 and 2 in Fruitchains and Subchains. We set the maximum number of blocks in a block race in Subchains $b_{\max} = 12$, and $l_{\max} = 24$ in the three other protocols to ensure a fair comparison.

Result 11: The subversion gain $S(\alpha, \sigma, V_{\mathrm{ds}})$ of Fruitchains and Subchains is larger than that of NC in our setting, while

that of RS is smaller.

Fruitchains and Subchains perform worse than NC for most α values. Fruitchains appears to achieve better performance when $\alpha = 0.45$ due to its MDP's limited action set. Indeed, if we truncate NC's and RS's action sets to the same as Fruitchains's, they outperform Fruitchains when $\alpha = 0.45$. The reasons of Subchains's and Fruitchains's unsatisfactory performance are similar to those of their $I(\alpha)$. As blocks in Fruitchains and weak blocks in Subchains have no reward, withholding them is risk-free. More audacious block-withholding behaviors result in higher expected double-spending reward regardless of how small R_{ds} is.

RS achieves better double-spending resistance than NC, and sometimes even achieves the ideal value 0, because the attacker has to balance the potential gain of double-spending and the potential loss in block rewards. When the risk outweighs the benefit, the attacker follows the compliant strategy.

2) Subversion bounty: To further evaluate a protocol's double-spending resistance, we define the *subversion bounty* $R_{\mathrm{sb}}(\alpha, \sigma)$ as the minimum R_{ds} that causes a rational attacker to deviate from the compliant strategy. We only compute $R_{\mathrm{sb}}(\alpha, \sigma)$ for NC and RS as $R_{\mathrm{sb}}(\alpha, \sigma) \equiv 0$ in the two other protocols. We choose $\gamma = 0.95$ rather than 1, because in the latter case, the attacker never follows the compliant strategy in NC, as every attacker block can orphan an honest block without any risk. The results are shown in Fig. 7.

Result 12: Raising σ drastically increases R_{sb} for weak attackers, but it is less effective for strong attackers.

Strong attackers can often find more than one block in a row, allowing them to initiate double-spending for less rewards.

Result 13: $R_{\mathrm{sb}}(\alpha, \sigma)$ decreases superlinearly with α.

The subversion bounty provides some guidance for merchants to choose the maximum value received in a block and the number of confirmations, based on the estimated attacker ability and the consensus protocol.

VII. Censorship Susceptibility Analysis

A. Modeling Censorship Susceptibility

The censorship susceptibility MDPs are different from incentive compatibility MDPs in their reward calculation. Here, the attacker's reward in a step is calculated as the compliant miners' loss O_c due to the attack. In NC, O_c is defined as the number of orphaned honest blocks. In Fruitchains, the attacker receives all compliant miners' fruit rewards if the attacker wins a block race no shorter than T_o. In RS, the attacker receives one block reward for each honest block rendered invisible and half of a block reward for each visible honest block with a competitor. In Subchains, the attacker receives $Ratio_{\mathrm{w2b}}$ units of rewards for each invalidated honest block.

B. Evaluation Results

The protocols' $C(\alpha)$ are computed with the following parameters. Three γ values are considered: 0, 0.5 and 1, with the exception that our Fruitchains MDP does not support $\gamma = 0.5$. We set $b_{\max}$ in Subchains as 20 and $l_{\max} = 40$ in the three other protocols to ensure a fair comparison. We truncate

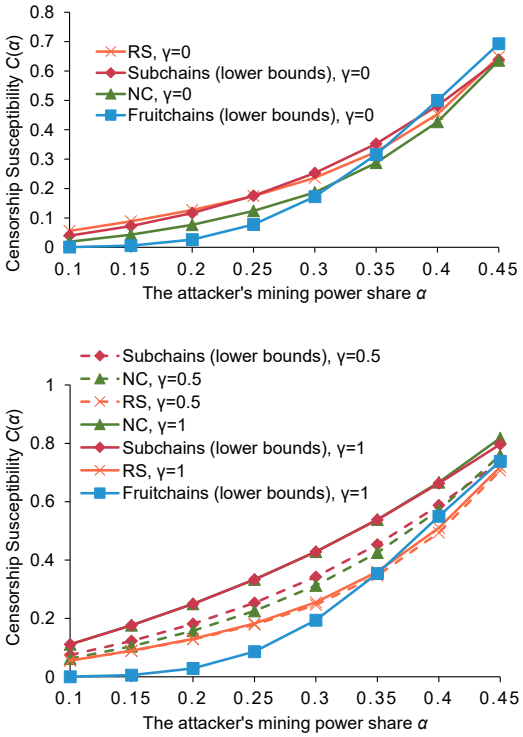

Fig. 8. Censorship susceptibility $C(\alpha)$ of four protocols, $l_{\max} = 40$. We put $\gamma = 0.5$ and $\gamma = 1$ in the same chart to save space. Larger number indicates worse performance.

a field representing the attacker's own fruits in Fruitchains MDP to enable larger values for $l_{\max}$, as these fruits do not contribute to the censorship attack. Other parameters are the same with our subversion gain evaluation. The results are listed in Fig. 8.

Result 14: Subchains's $C(\alpha)$ performs worse than NC, whereas Fruitchains performs better. RS's $C(\alpha)$ is worse than NC when $\gamma = 0$, but better when $\gamma \geq 0.5$.

Subchains performs worse than NC for all parameter sets with $\alpha < 0.45$ and $\gamma < 1$. When $\gamma = 1$, its performance is almost identical to that of NC. The reason for Subchains's poor performance in $C(\alpha)$ is similar to that of $I(\alpha)$. RS performs worse than NC when $\gamma = 0$ because in NC, the attacker cannot orphan an honest block with just one attacker block in a block race, whereas in RS, the attacker block can "loot" half of a block reward from its honest competitor. Fruitchains performs the best for all $\alpha \leq 0.3$ because in Fruitchains, the attacker cannot invalidate any honest fruit without winning a block race of length T_o, which is difficult for weak attackers.

An interesting fact is that when $\alpha \geq 0.4$, RS's $C(\alpha)$ outperforms that of Fruitchains, due to their different gap definitions. In Fruitchains, winning a block race with at least T_o blocks invalidates all honest fruits mined in the current block race, as their gaps are calculated from their pointer blocks, which are either "outdated"—mined before the current block race, or invalidated—not in the main chain. On the other hand, RS's gap is calculated from an uncle's own height,

therefore when the attacker wins a long block race, the last several honest blocks may still be referred to in the blockchain as valid uncles, splitting the attacker's rewards.

Result 15: Fruitchains's and RS's gap definitions perform better in terms of censorship resistance facing weak and strong attackers, respectively.

VIII. Security Trade-offs in Attack Resistance

A. Security vs. Performance

Our results confirm two security-performance trade-offs. First, longer confirmation delay contributes to better attack resistance, as shown in Result 6, 8, and our subversion bounty analysis. Second, higher bandwidth consumption, if properly utilized, strengthens the system by reducing the attacker's "lucky" space of gambling, as shown in Result 7. Moreover, our model quantifies the influence of each parameter on the protocols' attack resistance, allowing practitioners to choose these parameters according to their use cases.

B. "Rewarding the Bad" vs. "Punishing the Good"

None of the protocols we have studied successfully defends against all three attacks. Their weaknesses are not protocol-specific, but inherent to their technical approaches. Reward-all protocols improve censorship resistance by increasing the difficulty to invalidate other miners' rewards, at the price of removing the risk to fork the blockchain, thus encouraging double-spending attacks. Punishment protocols improve selfish mining and double-spending resistance by discouraging malicious behaviors, at the price of lowering the attacker's difficulty to damage the compliant miners' income, thus facilitating censorship. Reward-lucky protocols, contrary to their designers' intention, allow the attacker to invalidate the compliant miners' "lucky" blocks with the attacker's "unlucky" units in a risk-free manner, leaving them more vulnerable to all three attacks. We conclude that none of the three approaches can improve the security of PoW against three major attacks; they only offer different trade-offs in resistance. In other words, to date, no protocol achieves better resistance than NC in defending all three attacks.

We further summarize these weaknesses into a dilemma between "rewarding the bad" and "punishing the good", which roots in information asymmetry we identified in Sect. IV-D. Recall that due to this asymmetry, when the blockchain is forked, the protocol is unable to distinguish whether a contentious unit, be it a block, fruit or weak block, is a product of compliant or malicious behavior. As a result, if all contentious units are rewarded or punished equally, either "the bad" are rewarded, as in reward-all protocols, or "the good" are punished, as in punishment protocols. Selectively rewarding some contentious units without solving information asymmetry, as in reward-lucky protocols, usually increases the vulnerability to malicious manipulation, allowing both undesirable consequences to happen. This dilemma reveals that it is difficult, if not impossible, to defend against all three attacks with just a novel reward distribution policy.

IX. Discussion

A. Future Directions for PoW Protocol Designs

First, we highlight an empirical lesson summarized from our findings: complexity is the enemy of security. As demonstrated by our results, despite the simplicity of NC, to date there is no protocol that surpasses NC in all our security metrics when the attacker has no network propagation advantage. The seemingly more sophisticated later designs, contrary to their own claims, not only invite new attack strategies, but also complicate the analysis. In fact, some protocols are so complicated that their vulnerabilities could only be revealed through our MDP modeling.

As we have identified the cruxes of existing designs' unsatisfactory chain quality and attack resistance as their unrealistic and inconsistent security assumptions and the dilemma between "rewarding the bad" and "punishing the good", respectively, we present our suggestions on more secure PoW designs in the following two directions, accordingly.

1) Introducing and realizing practical assumptions to raise the chain quality: Such assumptions may include:

- *Awareness of network conditions.* Knowledge on whether the network is partitioned and the slowest block propagation time allows the participants to identify block withholding behaviors with a higher level of confidence. This information helps distinguish between honest and attacker blocks, and thus it contributes to raising chain quality.
  In the real world, well-established techniques from distributed databases can help to detect network partitions. The block propagation delay can be estimated from various measurement data, such as the current orphan rate [55], which are locally available or accessible from multiple online sources [56],
- *A loosely synchronized clock.* With a loosely synchronized clock, participants can use the gap between a block's receiving time and its timestamp as an indicator of malicious behaviors. This indicator could help to further raise the chain quality in combination with the previous assumption. Note that the assumption of a roughly accurate clock is necessary for all PoS protocols and is inherent to NC, as Bitcoin adjusts the block difficulty and the block reward according to the block timestamps reported by the miners.
- *Responsible parties with large deposits or public real-world identities.* The absence of legislation in permissionless blockchains is not in favor of security. This situation can be mitigated by demanding a large deposit before performing certain actions to increase the amount of penalty, or limiting these actions to parties with publicly verified real-world identities in order to put their reputation at stake.

Realizing these assumptions requires continuous work from researchers and developers as these seem to be necessary preconditions to improve the chain quality.

2) Outsourcing liability to raise attack resistance:

- *Introducing additional punishment rules.* The unfair rewards go to the malicious miners can be balanced with additional punishment. This approach demands that cryptographic

proofs of the malicious behaviors are embedded in the blockchain. For example, accountable assertions can be used to deter double-spending [57]. Designing such proofs for censorship attacks is an interesting research direction.

- *Relying on "layer 2" protocols to protect against specific attacks.* This approach reduces the consensus protocol's pressure on defending against certain attacks. For example, as Bitcoin's layer 2 solution, lightning network [58] guarantees double-spending resistance for its transactions, requiring the underlying consensus protocol to resist against selfish mining and censorship attacks.

B. Future Directions for PoW Protocols' Security Analyses

Three common pitfalls in existing security analyses prevent these vulnerabilities from being discovered in the first place:

- *Limiting the analysis to only one attack strategy.* Our work shows that such analysis is far from sufficient: protocol-specific rules often inspire new attack strategies, causing more damage than the generic strategy analyzed by the designer. Typical examples include SHTB's "smallest hash first" rule that inspires a "withhold when the hash is small enough" strategy and Subchains's "weak block counts in chain length" rule that inspires a "withhold weak blocks to invalidate honest blocks" strategy. In particular, given the recent advancement of artificial intelligence, we can expect future attackers to be equipped with more sophisticated strategies. Therefore, a solid protocol design calls for a formal, rather than a heuristic, security analysis.
- *Limiting the analysis against just one type of attacker incentive.* The blockchain ecosystem results in complex interactions between attackers and other players: an attacker may focus on short-term rewards, as in double-spending attacks, or risk short-term rewards for higher future returns, as in selfish mining, or even sacrifice all rewards to cause damage on other players, as in censorship attacks. This complexity, together with the multifunctional nature of blockchains, demands the security evaluation to be more comprehensive in terms of attacker incentive. Nevertheless, existing analyses typically focus on short-term reward seekers, leaving the protocol vulnerable to attackers with the two other incentives. The problem is more prominent for permissionless designs, where transactions are processed by anonymous parties, who abide by the protocol only out of their will and interests as defined by themselves. The lack of outside-the-blockchain negative consequences, especially legislative ones, opens the door for various attacker incentives which need to be taken into account.
- *Proving the system's security within an unrealistic parameter range.* Even if the security proofs give solid results, it is unclear whether the system is secure in a more realistic parameter range. For example, we reveal that Fruitchains is susceptible to selfish mining and double-spending attacks if the confirmation delay is shortened to more reasonable values. Therefore, we argue that future security analyses should depart from real-world parameters to provide more objective and meaningful results.

As demonstrated in this research, analyzing protocol security with artificial intelligence techniques has the following three-fold advantage. First, it simplifies the analysis with well-established algorithms, which enables us to analyze protocols more complicated than NC. Second, it allows accurate evaluation of the parameter choices. Third, these techniques can compute the attacker's optimal strategies, allowing designers to gain direct insights and iteratively improve their designs.

Note that, although vulnerability identification is simplified, it is more difficult to prove that a protocol resists against an attack with these techniques. Security cannot be claimed without proving that the strategy space used to compute the utility covers all rational strategies.

X. Related Work

Most research analyzing PoW protocol security focuses on NC [3], [51], [52], [59]–[62]. To the best of our knowledge, this paper presents the first cross-protocol multi-metric blockchain security evaluation.

Modeling a consensus protocol as a Markov process allows researchers to quantify the attacker's optimal utility with well-studied algorithms. Specifically, Gervais et al. study the selfish mining and double-spending resistance of NC with different parameters [31]. Zhang and Preneel evaluate the security of Bitcoin Unlimited, a Bitcoin scaling proposal [26]. Kiffer et al. [63] analyze Chainweb's and GHOST's *consistency*, namely whether all compliant parties share the same ledger, regardless of whether the ledger is biased by an attacker.

XI. Conclusion

Since the introduction of Bitcoin, new PoW designs emerge on a daily basis from both industry and academia. However, technology advancement cannot be simply measured by the number of protocols, but only by convincing improvements in performance or security. Unfortunately, the security of most of these alternative protocols remains self-claimed, and many of them seem to share similar vulnerabilities. To address this situation, this paper systematically analyze the security of seven most representative and influential alternative designs. However, our results show that none of these designs outperform NC in terms of either the chain quality or attack resistance in all scenarios. We identify the roots of their unsatisfactory performance as PoW protocols' unrealistic assumptions and information asymmetry between the compliant miners and the attacker. Moreover, we discover a considerable number of protocol-specific attacks and quantify two security-performance trade-offs with finer granularity. These results allow us to pinpoint some promising directions towards more secure PoW protocol designs and more solid security analysis.

Acknowledgements

This work was supported in part by Blockstream, the Flemish government imec ICON BoSS project, and the Research Council KU Leuven: C16/15/058. We would like to thank Yonatan Sompolinsky, Andrew Miller, Kaiyu Shao, Pieter Wuille, Gregory Maxwell, Adam Back and the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] mapofcoins. (2018) Map of coins: BTC map. [Online]. Available: http://mapofcoins.com/bitcoin

[2] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[3] J. A. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol: Analysis and applications," in *EUROCRYPT*, 2015, pp. 281–310.

[4] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *Financial Cryptography and Data Security*, 2016, pp. 515–532.

[5] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 305–320.

[6] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454.

[7] L. Bahack, "Theoretical Bitcoin attacks with less than half of the computational power (draft)," *arXiv preprint arXiv:1312.7013*, 2013.

[8] Ethereum white paper: Modified Ghost implementation. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper♯modified-ghost-implementation

[9] E. Heilman, "One weird trick to stop selfish miners: Fresh Bitcoins, a solution for the honest miner." Cryptology ePrint Archive, Report 2014/007, 2014, https://eprint.iacr.org/2014/007.

[10] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, "Inclusive block chain protocols," in *Financial Cryptography and Data Security*, 2015, pp. 528–547.

[11] P. R. Rizun, "Subchains: A technique to scale Bitcoin and improve the user experience," *Ledger*, 2016. [Online]. Available: https://www.ledgerjournal.org/ojs/index.php/ledger/article/view/40

[12] S. D. Lerner. (2015) DECOR+HOP: A scalable blockchain protocol. [Online]. Available: https://scalingbitcoin.org/papers/DECOR-HOP.pdf

[13] R. Zhang and B. Preneel, "Publish or Perish: A backward-compatible defense against selfish mining in Bitcoin," in *CT-RSA 2017: The Cryptographers' Track at the RSA Conference*, 2017, pp. 277–292.

[14] Y. Sompolinsky, Y. Lewenberg, and A. Zohar. (2016) SPECTRE: Serialization of proof-of-work events: Confirming transactions via recursive elections. [Online]. Available: https://eprint.iacr.org/2016/1159.pdf

[15] I. Bentov, P. Hubácek, T. Moran, and A. Nadler, "Tortoise and Hares consensus: the Meshcash framework for incentive-compatible, scalable cryptocurrencies," *IACR Cryptology ePrint Archive*, 2017.

[16] Y. Sompolinsky and A. Zohar, "PHANTOM: A scalable blockdag protocol," *IACR Cryptology ePrint Archive*, 2018. [Online]. Available: https://eprint.iacr.org/2018/104.pdf

[17] S. D. Lerner. (2015) RSK white paper overview. [Online]. Available: https://zh.scribd.com/document/371006520/RSK-White-Paper-Overview

[18] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing Bitcoin security and performance with strong consistency via collective signing," in *Proc. 25th conference on USENIX Security Symposium*, 2016.

[19] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," *Proc. 38th IEEE Symposium on Security and Privacy*, 2018.

[20] R. Pass and E. Shi, "Fruitchains: A fair blockchain," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, ser. PODC '17. ACM, 2017, pp. 315–324. [Online]. Available: http://doi.acm.org/10.1145/3087801.3087809

[21] P. Camacho and S. D. Lerner. (2016) DECOR+LAMI: A scalable blockchain protocol. [Online]. Available: https://scalingbitcoin.org/papers/DECOR-LAMI.pdf

[22] G. Bissias and B. N. Levine, "Bobtail: A proof-of-work target that minimizes blockchain mining variance (draft)," *CoRR*, vol. abs/1709.08750, 2017. [Online]. Available: http://arxiv.org/abs/1709.08750

[23] W. Martino, M. Quaintance, and S. Popejoy. (2018) Chainweb: A proof-of-work parallel-chain architecture for massive throughput. [Online]. Available: http://kadena.io/docs/chainweb-v15.pdf

[24] C. Natoli and V. Gramoli, "The balance attack against proof-of-work blockchains: The R3 testbed as an example," 2016.

[25] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain." *IACR Cryptology ePrint Archive*, vol. 2016, p. 545, 2016.

[26] R. Zhang and B. Preneel, "On the necessity of a prescribed block validity consensus: Analyzing bitcoin unlimited mining protocol," in *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. ACM, 2017, pp. 108–119.

[27] H. Nguyen. (2018) Proof-of-stake & the wrong engineering mindset. [Online]. Available: https://medium.com/@hugonguyen/proof-of-stake-the-wrong-engineering-mindset-15e641ab65a2

[28] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains," *CoRR*, vol. abs/1711.03936, 2017. [Online]. Available: http://arxiv.org/abs/1711.03936

[29] J. Brown-Cohen, A. Narayanan, C.-A. Psomas, and S. M. Weinberg, "Formal barriers to longest-chain proof-of-stake protocols," *arXiv preprint arXiv:1809.06528*, 2018.

[30] A. Miller. (2013) Feather-forks: enforcing a blacklist with sub-50[Online]. Available: https://bitcointalk.org/index.php?topic=312668.0

[31] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proc. the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. ACM, 2016, pp. 3–16. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978341

[32] Y. Sompolinsky and A. Zohar, "Bitcoin's security model revisited," *arXiv preprint arXiv:1605.09193*, 2016.

[33] ——, "Secure high-rate transaction processing in Bitcoin," in *Financial Cryptography and Data Security*, 2015, pp. 507–527.

[34] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for Bitcoin and cryptocurrencies," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2015, pp. 104–121.

[35] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of Bitcoin without the block reward," in *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. ACM, 2016, pp. 154–167. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978408

[36] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*. Princeton University Pres, 2016.

[37] Decred Developers. (2018) Decred - autonomous digital currency. [Online]. Available: https://www.decred.org/

[38] J. Bonneau, "Why buy when you can rent? bribery attacks on Bitcoin-style consensus," in *BITCOIN workshop, Financial Cryptography and Data Security*. Springer, 2016.

[39] D. Meshkov, A. Chepurnoy, and M. Jansen, "Revisiting difficulty control for blockchain systems," in *DPM/CBT@ESORICS 2017*, 2017. [Online]. Available: https://eprint.iacr.org/2017/731.pdf

[40] I. Eyal, "The miner's dilemma," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2015, pp. 89–103.

[41] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 195–209.

[42] I. Tsabary and I. Eyal, "The gap game," in *Proceedings of the 11th ACM International Systems and Storage Conference*. ACM, 2018.

[43] R. Pass and E. Shi, "Thunderella: Blockchains with optimistic instant confirmation," in *Advances in Cryptology – EUROCRYPT 2018*. Springer International Publishing, 2018, pp. 3–33.

[44] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[45] T. Duong, A. Chepurnoy, L. Fan, and H.-S. Zhou, "Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake," in *Proceedings of the 2Nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*, ser. BCC '18. ACM, 2018, pp. 1–13. [Online]. Available: http://doi.acm.org/10.1145/3205230.3205233

[46] V. Buterin. (2014) Ethereum: A next-generation smart contract and decentralized application platform. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[47] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, 2016, pp. 45–59. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal

[48] (2018) Waves platform. [Online]. Available: https://wavesplatform.com/
[49] (2018) Aeternity blockchain. [Online]. Available: https://aeternity.com/
[50] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www.opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.
[51] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 643–673.
[52] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, "Blockchain mining games," in *Proceedings of the 2016 ACM Conference on Economics and Computation*. ACM, 2016, pp. 365–382.
[53] K. Liao and J. Katz, "Incentivizing blockchain forks via whale transactions," in *Financial Cryptography and Data Security*. Springer International Publishing, 2017, pp. 264–279.
[54] Wolfram Research, Inc. (2018) Newton-Pepys problem. [Online]. Available: http://mathworld.wolfram.com/Newton-PepysProblem.html
[55] C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *13th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2013.
[56] Bitcoin stats - data propagation. [Online]. Available: http://bitcoinstats.com/network/propagation/
[57] T. Ruffing, A. Kate, and D. Schröder, "Liar, liar, coins on fire!: Penalizing equivocation by loss of Bitcoins," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 219–230. [Online]. Available: http://doi.acm.org/10.1145/2810103.2813686
[58] J. Poon and T. Dryja, "The Bitcoin lightning network: Scalable off-chain instant payments," 2016.
[59] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin backbone protocol with chains of variable difficulty," in *Annual International Cryptology Conference*. Springer, 2017, pp. 291–323.
[60] J. A. Garay, A. Kiayias, and G. Panagiotakos, "Proofs of work for blockchain protocols," Cryptology ePrint Archive, Report 2017/775, Tech. Rep., 2017.
[61] P. Wei, Q. Yuan, and Y. Zheng, "Security of the blockchain against long delay attack," in *Advances in Cryptology–ASIACRYPT 2018*, 2018.
[62] T. Duong, A. Chepurnoy, and H.-S. Zhou, "Multi-mode cryptocurrency systems," in *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts*. ACM, 2018, pp. 35–46.
[63] L. Kiffer, R. Rajaraman *et al.*, "A better method to analyze blockchain consistency," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 729–744.

APPENDIX A
SHTB MDP DESIGN

A. Properties of Deterministic Tie-Breaking Protocols

We can simplify the state representation in these protocols by omitting two kinds of information that do not affect the miners' choices of parent blocks. First, we do not need to encode the mining history, as "latecomer" blocks can still win a tie. Second, we do not need to explicitly encode how many attacker chain blocks are published, as it can be deduced from the public chain length l_c. As compliant miners always work on the same chain in deterministic tie-breaking protocols, if the attacker publishes enough blocks so that the compliant miners switch to the attacker chain, the public chain is abandoned and l_c is updated to zero; otherwise as long as $l_c > 0$ we can safely assume the compliant miners are working on the public chain, thus different numbers of published attacker chain blocks make no difference to all miners. This analysis also shows that compliant miners always work on the public chain in deterministic tie-breaking protocols.

B. State Space

We use l_a and l_c to denote the lengths of the attacker chain and the public chain, respectively, excluding their common blocks. The hash region of the public chain tip is denoted as Hash_c. If we normalize the space of valid block hash to $[0, 1)$ and split it into 10 regions, $\mathit{Hash}_c = h$ means the hash resides in $[0.1h, 0.1(h+1))$, where h is the region number, an integer ranges from 0 to 9. Hash_a^1 and Hash_a^2 represent the hash regions of the last and the second last attacker chain blocks, respectively. When $l_a \geq l_c > 0$, *tie* denotes whether the public chain tip is smaller than its attacker chain competitor. It has two possible values: aWin, meaning the attacker chain competitor is smaller, and aLose, meaning the public chain tip is smaller.

The state representation differs according to the length difference of the chains: (1) *When* $l_a < l_c$, a state is represented as a 3-tuple (l_a, l_c, Hash_c). As the public chain is longer, the compliant miners will not mine on the attacker chain, thus there is no need to encode Hash_a^1 and Hash_a^2. Hash_c is encoded in case the attacker catches up from behind. (2) *When* $l_a = l_c$, a state is a 3-tuple (l_a, l_c, *tie*). When *tie* $= \mathrm{aWin}$, the attacker can orphan the public chain by publishing the entire attacker chain. (3) *When* $l_a = l_c + 1$, a state is a 4-tuple (l_a, l_c, *tie*, Hash_a^1). When *tie* $= \mathrm{aWin}$, the attacker can orphan the public chain by publishing until the l_c-th attacker block; otherwise the attacker needs to publish the entire chain to win the race. When $l_c = 0$, *tie* is undefined, denoted as $\emptyset$. (4) *When* $l_a > l_c + 1$, a state is a 4-tuple (l_a, l_c, Hash_a^1, Hash_a^2). Instead of encoding the hash regions of all attacker blocks in the leading part, we only encode the last two. The attacker is not allowed to orphan the public chain by winning a tie when more than one block ahead, which favors the compliant miners.

C. Actions

The attacker can choose from four actions:

Adopt. Give up the attacker chain and mine on the public chain. This action is always available.

OverrideWithTie. Publish until the l_c-th attacker block to orphan the public chain, and keep mining on the attacker chain after publication. Available when *tie* $= \mathrm{aWin}$.

OverrideWithMore. Publish until the $(l_c + 1)$-th attacker block to orphan the public chain, and keep mining on the attacker chain. Available when $l_a > l_c$.

Wait. Do not publish anything and keep mining on the attacker chain. Always available.

We do not claim that this action set covers all optimal actions. It is possible that in certain states, the optimal action is to publish more than l_c+1 blocks, which is not in our action set. This constrained attacker action set favors the compliant miners.

An interesting implication from this action set definition is that we can assume the attacker always mines on the attacker chain. *Adopt* can be considered as working on an empty attacker chain. Note that this does not exclude the compliant strategy from the strategy space. The compliant strategy is equivalent to choosing *Adopt* when the last block is honest

and choosing *OverrideWithMore* when the last block is the attacker's. This implication applies to all our MDPs.

D. Reward Allocation and State Transition

The compliant miners get $R_c = l_c$ only after *Adopt*. The attacker gets $R_a = l_c$ or $l_c + 1$ after *OverrideWithTie* or *OverrideWithMore*, respectively. After each of these three actions, information regarding blocks that are permanently abandoned or accepted by both miners will be cleared in the new temporary state. No reward is allocated after *Wait*.

When a new block is mined, it has equal probability to reside in every hash region. For example, when there are 10 valid hash regions, the probability that the compliant miners find the next block in region 3 is $(1-\alpha)/10$. Assuming the new block's hash region is $\mathit{Hash}^{\text{new}}$, if the new block's chain is longer than its competitor, $\mathit{Hash}^{\text{new}}$ will be encoded in the next state as Hash_a^1 or Hash_c, depending on the miner. Before replacing a non-empty Hash_a^1, the old Hash_a^1 is stored as the new Hash_a^2: $\mathit{Hash}_a^{2,\text{new}} = \mathit{Hash}_a^1$. If $l_a = l_c - 1$ in the post-publishing temporary state and the new block is mined by the attacker, we have *tie* $=$ aWin in the new state if $\mathit{Hash}^{\text{new}}$ is smaller than the previous Hash_c or *tie* $=$ aLose if $\mathit{Hash}^{\text{new}}$ is equal to or bigger than the previous Hash_c. As an example, if $l_a = l_c - 1$ and $\mathit{Hash}_c = 3$ in the post-publishing state, the probability that the next state is $(l_a+1, l_c, \mathit{tie}^{\text{new}} = \text{aWin})$ is $\alpha \times 3/10$, as the attacker can only win the tie with $\mathit{Hash}^{\text{new}} = 0, 1$, or 2; the probability that the next state is $(l_a + 1, l_c, \mathit{tie}^{\text{new}} = \text{aLose})$ is $\alpha \times (10-3)/10$. The same rule is followed for updating *tie* when the public chain is catching up from behind the attacker chain.

Appendix B
UDTB MDP Design

A. State Space

As the probability of winning a tie is fixed to 50%, there is no need to encode the hashes of the latest blocks. Therefore, we can simplify the state representation of the previous MDP as follows. (1) *When* $l_a < l_c$ *or* $l_c = 0$, a state is a two-tuple (l_a, l_c). (2) *When* $l_a \geq l_c > 0$, a state is a 3-tuple (l_a, l_c, *tie*).

B. Actions

The action set is the same with the previous MDP. According to the action set completeness proof in Appendix A of [4], this set covers all rational actions. Note that the proof is not applicable to SHTB as blocks in SHTB are not interchangeable: a block with smaller hash is more likely to win a tie.

C. Reward Allocation and State Transition

The reward allocation mechanism is identical to that of the previous MDP. The state transition rules for updating l_a and l_c are straightforward, hence we only highlight the updating rule for *tie* here. The new *tie* is different from the previous one in three occasions. First, when $l_a = l_c - 1$ in the post-publishing temporary state and the new block is mined by the attacker, the transition probability to the new state $(l_a+1, l_c, aWin)$ is $\alpha/2$ and the same to $(l_a+1, l_c, aLose)$. Second, when $l_a > l_c$ in the post-publishing state and the new block is mined by the compliant miners, the transition probability to the new state $(l_a, l_c+1, aWin)$ is $(1-\alpha)/2$ and the same to $(l_a, l_c+1, aLose)$. At last, when $l_a = l_c$ in the post-publishing state and the new block is mined by the compliant miners, *tie* is cleared and the transition probability to the new state (l_a, l_c+1) is $1-\alpha$. In all other situations, *tie* remains unchanged.

Appendix C
The Fruitchains MDP Design

Unlike in previous MDPs where a block is found at the end of each step, in the Fruitchains MDP, each step ends with the discovery of a *unit*, which might be a fruit or a block.

A. State Space

Encoding each fruit's pointer block in a state is computationally infeasible due to the potentially large number of fruits. Therefore, we split all fruits into three groups and deal with them separately: (1) attacker fruits mined before the T_o-th attacker block; (2) attacker fruits mined after the T_o-th attacker block; (3) honest fruits. As the attacker knows which block is the consensus block, it is rational that fruits in group (1) point to the consensus block, so that they can be published before expiration and embedded in both chains. As these fruits always receive rewards, we can issue their rewards the moment they are found, and forget them in the next state. Fruits in group (2) gain rewards if and only if the attacker wins the block race, because otherwise the pointer blocks of these fruits are invalidated. Fruits in group (3) lose the rewards when the attacker wins the block race with at least T_o blocks, either because their pointer blocks are invalidated or because their gaps exceed T_o. For all other scenarios, either the attacker loses or wins with less than T_o blocks, we assume all honest fruits receive rewards. This setting favors the compliant miners, as the attacker may still invalidate some honest fruits when winning with less than T_o blocks: either the honest fruits expire after the current block race, as their pointers are before the consensus block; or the attacker wins the following block races and obtains T_o consecutive main chain blocks eventually, causing the honest fruits mined in the first block race to expire.

A state is represented as a 4-tuple (l_a, l_c, f_c, *isLastHB*) when $l_a < T_o$, or a 5-tuple (l_a, l_c, f_c, $f_a^{\text{after T}_o}$, *isLastHB*) when $l_a \geq T_o$, where f_c denotes the number of honest fruits, $f_a^{\text{after T}_o}$ denotes the number of attacker fruits mined after the T_o-th attacker block. A Boolean value *isLastHB* stores whether the last unit is an honest block.

B. Actions

The attacker can choose from three actions:

Adopt. Give up the attacker chain. Same as previous MDPs.

Override. Publish all fruits and blocks to orphan the public chain. When $\gamma = 0$, this action is only available when $l_a \geq l_c + 1$; when $\gamma = 1$, this action is also available when $l_a = l_c$ and *isLastHB* $=$ true. Due to the complexity of Fruitchains, we do not consider other γ values.

Wait. Keep mining on the attacker chain. Same as previous MDPs.

This limited set of actions does not allow pre-mining. Namely, the attacker cannot publish some blocks and fruits, and carry other secret units to the next block race.

C. Reward Allocation and State Transition

A valid fruit receives $1/\mathit{Ratio}_{\mathrm{f2b}}$ so that on average one unit of reward is issued per block. The attacker receives one fruit reward for each fruit mined before the T_o-th attacker block. If the attacker chooses *Override* when $l_a < T_o$ or *Adopt*, the compliant miners receive f_c fruit rewards. If the attacker chooses *Override* when $l_a \geq T_o$, the compliant miners receive nothing and the attacker receives $f_a^{\mathrm{after\,T_o}}$ fruit rewards. All settled fruits and blocks are cleared in the new temporary state.

The new unit found at the end of a step can be an attacker block, an attacker fruit, an honest block or an honest fruit, with probability $\alpha/(1+\mathit{Ratio}_{\mathrm{f2b}})$, $\alpha \cdot \mathit{Ratio}_{\mathrm{f2b}}/(1+\mathit{Ratio}_{\mathrm{f2b}})$, $(1-\alpha)/(1+\mathit{Ratio}_{\mathrm{f2b}})$, $(1-\alpha)\cdot \mathit{Ratio}_{\mathrm{f2b}}/(1+\mathit{Ratio}_{\mathrm{f2b}})$, respectively. For example, when $\alpha = 1/3$ and $\mathit{Ratio}_{\mathrm{f2b}} = 2$, the probabilities of finding an attacker block, an attacker fruit, an honest block and an honest fruit are $2/9$, $1/9$, $4/9$ and $2/9$, respectively. When the latest unit is an honest block, *isLastHB* $= \mathrm{true}$, otherwise *isLastHB* $= \mathrm{false}$.

Appendix D
Reward-Splitting Protocol MDP Design

It is never optimal for the attacker to hide a block forever, as a late publication still gains at least half of a block reward. Similarly, the attacker blocks never embed honest uncles, hoping that they could be rendered invisible.

A. State Space

An honest block of height h becomes invisible when the main chain blocks between height h and $h+T_o-1$ are all mined by the attacker. Therefore, our state representation needs to encode previous consecutive block races won by the attacker up to T_o-1 height values. We encode this history information as *history*, a binary string of at most T_o-1 bits. The length of *history* represents the number of consecutive attacker main chain blocks. Each bit indicates whether the attacker block has an honest competitor: 0 means no, 1 means yes. The least significant bit represents the blockchain status at the consensus block's height, denoted as h_{con}, and the second least significant bit represents that of height $h_{\mathrm{con}}-1$. Other bits follow similar definitions. A substring from height h_1 to h_2 where $h_1 \leq h_2$ is denoted as $\mathit{history}[h_1:h_2]$, thus *history* is equivalent to $\mathit{history}[h_{\mathrm{con}}-T_o+2:h_{\mathrm{con}}]$. When $h_1 > h_2$ the substring is empty. We do not need to encode blocks at height $h_{\mathrm{con}}-T_o+1$ and lower, as their rewards are settled along with the current consensus block. Neither do we need to encode whether a leading zero is an attacker block without an honest competitor or a block race won by the compliant miners, as in both cases the rewards are settled already, which will be further explained when describing the reward allocation. The number of 1s in the substring is denoted as $\sum \mathit{history}[h_1:h_2]$.

A state is represented as a 4-tuple (l_a, l_c, *fork*, *history*), where *fork* has three possible values. If there is an ongoing tie, namely the attacker chain is published until the l_c-th block and this block is published along with the latest honest block, *fork* $=$ active. Otherwise if the latest block is mined by the compliant miners, *fork* $=$ cLast; *fork* $=$ aLast if the attacker finds the last block.

B. Actions

There are T_o+2 possible optimal actions:

Adopt. Give up the attacker chain. Same as previous MDPs.

Wait. Keep mining on the attacker chain. Same as previous MDPs.

Match. Publish until the l_c-th attacker block to cause a tie, then keep mining on the attacker chain. Feasible when $l_a \geq l_c$ and *fork* $=$ cLast, namely the attacker has enough blocks to match the newly-mined honest block.

Override$_k$. Publish until the (l_c+k)-th attacker block to orphan the public chain, then keep mining on the attacker chain, where $1 \leq k \leq T_o-1$. Feasible when the attacker has enough blocks.

This action set covers all optimal actions. It is never optimal to publish the (l_c+T_o)-th attacker block, as the attacker can invalidate one more honest block without risking any block reward by deferring this attacker block's publication until the next honest block is mined.

C. Reward Allocation and State Transition

An attacker block is certain to receive the full reward if it has no competing honest block when published. Therefore, we issue block rewards to these "no competitor" attacker blocks the moment they are published. Consequently, the rewards of all 0s in *history* are settled before they enter *history*.

When choosing *Adopt*, the compliant miners receive $l_c - l_a$ full rewards for honest blocks without a competitor, and $(\sum \mathit{history} + l_a)/2$ for honest blocks with a competitor. The attacker receives $(\sum \mathit{history} + l_a)/2$ for the attacker blocks. We assume $l_a \leq l_c$ here, as otherwise *Override*$_1$ is clearly more profitable than *Adopt*. After *Adopt*, $\mathit{history}^{\mathrm{new}}$ is empty.

When choosing *Override*$_k$, the attacker receives two kinds of rewards. The first kind are for attacker blocks that have competitors but the competitors are pushed out of *history* after this action. We first append $1^{l_c}||0^k$, a string denotes the current block race, to the end of *history*, then truncate the resulted string to T_o-1 least significant bits. When $T_o-1 \geq l_c+k$, $\mathit{history}^{\mathrm{new}} = \mathit{history}[h_{\mathrm{con}}-T_o+2+l_c+k : h_{\mathrm{con}}]||1^{l_c}||0^k$. The attacker receives $\sum \mathit{history}[h_{\mathrm{con}}-T_o+2 : h_{\mathrm{con}}-T_o+1+l_c+k]$ for all 1s in the discarded *history* bits. Otherwise when $T_o-1 < l_c+k$, the attacker receives $\sum \mathit{history}+l_c+k-(T_o-1)$ for all 1s in *history* and the first $l_c+k-(T_o-1)$ attacker blocks in the current block race, as their competitors are invalidated, and $\mathit{history}^{\mathrm{new}} = 1^{T_o-1-k}||0^k$. The second kind of rewards are for the last k published attacker blocks, as they have no honest competitor.

No reward is allocated after *Wait* if *fork* $\neq$ active. There are two possible states after *Wait* if *fork* $\neq$ active, *Adopt* and

Override$_k$: either the next block is mined by the attacker on the attacker chain with probability α, or the next block is mined by the compliant miners on the public chain with probability $1-\alpha$. In the former case, *fork*$^{\text{new}} = \text{aLast}$; in the latter case, *fork*$^{\text{new}} = \text{cLast}$.

Unlike the previous actions, there are three possible states after *Wait* if *fork* $= \text{active}$ or *Match*. First, the attacker mines a block on the attacker chain with probability α. This is the only transition in the entire MDP where *fork*$^{\text{new}} = \text{active}$. Second, the compliant miners mine on the public chain with probability $(1-\alpha)(1-\gamma)$, *fork*$^{\text{new}} = \text{cLast}$. In the first two cases, no reward is allocated and *history*$^{\text{new}} =$ *history*. Third, the compliant miners mine on the attacker chain with probability $(1-\alpha)\gamma$. In this case, *history* is appended with 1^{l_c} and truncated until at most $T_o - 1$ bits. The attacker receives rewards for all 1s in the discarded history bits. The new state is $(l_a - l_c, 1, \text{cLast}, \textit{history}^{\text{new}})$.

APPENDIX E
SUBCHAINS MDP DESIGN

A. State Space

Similar to Fruitchains MDP, in Subchains MDP, each step ends with the discovery of a unit—either a block or a weak block. Based on our key observation in Sect. V-C, of the two mining sequences, only the leading unit sequence of the attacker chain, i.e., the units whose heights are larger than the public chain tip, needs to be encoded, as other bits are either adopted or abandoned as a whole. Therefore, we introduce two extra fields to facilitate state representation compression. First, *lead* denotes the attacker chain's leading unit sequence. Each bit in a string indicates whether the unit is a block or a weak block: 0 means a weak block, 1 means a block. The most significant bit represents the oldest unit in the chain, while the least significant bit presents the latest. Second, we encode the length difference between two chains as *diff*$_u$.

The state representation differs according to the length difference of the chains. (1) *When diff*$_u < 0$, a state is a 3-tuple $(b_a, b_c, \textit{diff}_u)$, where b_a and b_c denote the number of blocks in the attacker and the public chain, respectively. (2) *When diff*$_u = 0$, a state is a 4-tuple $(b_a, b_c, \textit{diff}_u, \textit{fork})$. Similar to *fork* in RS MDP, *fork* here denotes whether there is an ongoing tie, and if not, the miner of the last unit. There is no need to encode *fork* in the previous case as it is infeasible for the attacker to cause a tie. (3) *When diff*$_u > 0$, a state is a 5-tuple $(b_a, b_c, \textit{diff}_u, \textit{lead}, \textit{fork})$. For example, $(1, 3, 2, \text{“01”}, \text{aLast})$ means: the attacker chain and the public chain have one and three blocks, respectively; the attacker chain is two units longer than the public chain, of which the penultimate unit is a weak block, the last unit is a block mined in the last round.

B. Actions

The attacker can choose from four actions: *Adopt*, *Override*, *Match* and *Wait*. *Adopt* and *Wait* are the same with previous MDPs.

Match. Publish until the published attacker chain is of the same length with the public chain to cause a tie, then keep mining on the attacker chain. Feasible when *fork* $= \text{cLast}$ and *diff*$_u = 0, 1, 2,$ or 3. The requirement on *diff*$_u$ is because we set the maximum length of *lead* to three in order to further compress the state space. When *diff*$_u > 3$, *lead* only encodes the last three attacker units.

Override. When *diff*$_u = 1, 2$ or 3, publish until the published attacker chain is one unit longer than the public chain; when *diff*$_u > 3$, publish all attacker units except the last three.

This limited action set favors the compliant miners.

C. Reward Allocation and State Transition

We issue each block *Ratio*$_{\text{w2b}}$ units of rewards, so that on average each block or weak block receives one unit of reward. As both weak blocks and blocks contribute to the transaction confirmation, this “one reward per confirmation” rule is consistent with the reward allocation mechanisms of NC, Fruitchains and RS.

The compliant miners get $R_c = b_c \times \textit{Ratio}_{\text{w2b}}$ only after *Adopt*. After *Override*, the attacker gets rewards for all published attacker blocks, which is $R_a = (b_a - \sum \textit{lead})\textit{Ratio}_{\text{w2b}}$ when *diff*$_u > 3$ or *diff*$_u \leq 3$ and the highest order bit of *lead* is zero, or $R_a = (b_a - \sum \textit{lead} + 1)\textit{Ratio}_{\text{w2b}}$ when *diff*$_u \leq 3$ and the highest order bit of *lead* is 1. If the next unit is mined by the compliant miner on the attacker chain after *Wait* when *fork* $= \text{active}$ or *Match*, the attacker gets $R_a = (b_a - \sum \textit{lead})\textit{Ratio}_{\text{w2b}}$. After each of these actions, information regarding blocks and weak blocks that are permanently abandoned or accepted by both miners will be cleared in the new temporary state. No reward is allocated after *Wait* when *fork* $\neq \text{active}$.

There are four outcome states after *Wait* when *fork* $\neq \text{active}$, *Adopt* or *Override*, depending on the next unit. The new mining product can be an attacker block, an attacker weak block, an honest block or an honest weak block, with probability $\alpha/\textit{Ratio}_{\text{w2b}}$, $\alpha \cdot (\textit{Ratio}_{\text{w2b}} - 1)/\textit{Ratio}_{\text{w2b}}$, $(1-\alpha)/\textit{Ratio}_{\text{w2b}}$, $(1-\alpha) \cdot (\textit{Ratio}_{\text{w2b}} - 1)/\textit{Ratio}_{\text{w2b}}$, respectively. Meanwhile, after *Wait* when *fork* $= \text{active}$ or *Match*, the new honest unit might be mined on either chains, resulting in six outcome states. For example, the probability of an honest block mined on the attacker chain is $(1-\alpha)\gamma/\textit{Ratio}_{\text{w2b}}$.

We now describe how to get the new state from the temporary state after publication and the new unit. The rule for updating *fork* is identical to that of RS. If the next unit is honest, *diff*$_u$ decreases by one, otherwise it increases by one. If the next unit is a block, b_a or b_c increases by one according to the miner.