Resilience of Mining Pool Reward Schemes

Mustafa Doger Sennur Ulukus

Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
doger@umd.edu *ulukus@umd.edu*

Abstract

Mining pools share block rewards among their members according to a payout scheme. A pool member can increase its revenue by withholding the blocks it finds (block withholding) or by moving its hashpower in and out of the pool (pool hopping). Prior analyses rely on simplifying assumptions: withholding is analyzed under a global-reset version of the proportional scheme (GRP) that pools do not use, share-based schemes only under a fixed difficulty, and hopping only for a negligible miner. We analyze both attacks under the schemes used in practice: Pay-Per-Share (PPS), Pay-Per-Last-N-Shares (PPLNS), Proportional (PROP) and exponential-decay (EXP), taking the difficulty adjustment (DA) into account. For withholding, we propose Temporary Power Adjusting Withholding (T-PAW), which generalizes and improves known withholding attacks, and derive its exact revenues under each block-dependent scheme. We show that GRP overstates the gain under PROP by up to more than an order of magnitude and understates it under EXP. For PPS, the optimal attack is to withhold all blocks, which raises the revenue of an adversary with a fraction α of the hashpower from α to $\frac{\alpha}{1-\alpha}$ per epoch after the DA, at the expense of the pool operator. For hopping, we introduce Power Adjusting Hopping (PAH) for an adversary of any size and derive its revenues in closed form. PROP is most vulnerable to hopping and PPS to withholding by a large adversary, whereas EXP is vulnerable to both: a slow decay invites hopping and a fast decay invites withholding. PPLNS is the most robust. Monte Carlo simulations verify our results.

I. Introduction

Nakamoto consensus enables distributed participants in a network to reach agreement in a permissionless and trustless manner. By solving cryptographic puzzles, a process known as *mining*, participants, referred to as *miners*, become eligible to append a new block containing the embedded solution, termed *Proof-of-Work (PoW)*, to the chronological chain, collectively called a *blockchain*. Nakamoto incentivizes mining by rewarding miners for successfully extending the blockchain and broadcasting newly mined blocks to other participants [1]. The (honest) protocol prescribes miners to mine on top of the longest chain and share newly mined blocks immediately with their peers. When everyone follows the honest protocol, miners get rewards that are proportional (in expectation) to their fraction of hashpower in the system.

For a single miner, as the global hashrate grows, it becomes statistically harder to solve PoW puzzles, hence miners cooperate under pools and share the reward of the blocks they create. Pools distribute rewards of blocks according to their choice of payout schemes. Historically, pools utilized block-dependent reward mechanisms such as Proportional (PROP), Pay-Per-Last-N-Shares (PPLNS) and score-based such as Exponential Decay (EXP), under which block rewards are distributed among members upon the successful mining of a block. In these schemes, pool participants collectively bear the variance associated with block discovery times. In contrast, many modern pools adopt share-based reward mechanisms such as Pay-Per-Share (PPS) and Full-PPS (FPPS), under which members receive immediate compensation for each share submitted. Consequently, the risk of mining time variance is entirely borne by the pool operator [2].

Eyal and Sirer [3] demonstrated that Nakamoto consensus is vulnerable to incentive-based attacks: a miner can temporarily withhold the blocks it mines according to a selfish mining strategy and subsequently release them to replace those mined by other participants. Such an attack can increase the fraction of the blocks mined by the attacker and after the difficulty adjustment (DA), the attacker begins to accrue profits [4].

Similar to selfish mining, another adversarial mining strategy, called *Block Withholding* (BWH) attack, targeted at pool mining with block-dependent schemes, increases an attacker's share of rewards [5]. Later studies [6], [7] re-analyzed the attack under more general conditions using rigorous formalisms. In a nutshell, with a fraction of its mining power, an adversary joins a pool, withholds the valid blocks, i.e., full PoWs (fPoWs), it finds and submits

Topic	Prior work	This paper
BWH variants	FAW/PAW under GRP [9]–[11]	FAW/PAW/T-PAW under PPLNS, PROP, EXP; GRP is not reliable
Short-term gain	PAW under GRP [11]	Profit lag of T-PAW, PAH and AoA under every scheme
Hopping, PROP	Negligible miner, all-or-nothing [2], [21]	PAH for any adversary size and allocations
Hopping, EXP	Qualitative [2], observed in practice [22]	Closed-form analysis, trade-off in decay rate
Hopping, PPLNS	Hopping-proof [2]	Any adversary size and policy
PPS	Incentive compatible under fixed difficulty [25], [29]	Not incentive compatible under DA, optimal attack AoA

TABLE I: Prior work and the gaps addressed in this paper.

only the partial PoWs (pPoWs) that prove its work, thereby still receiving rewards from the blocks mined by the other pool members. By carefully allocating its power between honest and targeted pool mining, an adversary can increase its fraction of the rewards. In 2014, Eligius mining pool was subject to a BWH attack and lost 300 BTC [8].

There are two natural extensions of BWH: *Fork After Withholding* (FAW) [9] and *Power Adjusting Withholding* (PAW) [10]. Each extension offers progressively greater flexibility to the attacker, enabling increased adversarial profits. In FAW, the adversary releases the withheld block when an honest miner (outside the target pool) mines a block to cause a fork race [9]. PAW further improves on FAW by allowing the adversary to readjust its power allocation to pool mining after it finds a block, which increases the extra revenue up to 2.5 times that of FAW [10]. An in-depth analysis of PAW is given in [11] which derives PAW-related quantities in an explicit manner without the simplifying assumptions of [9], [10] related to fork races. Other works, including [12]–[18], study BWH, FAW and PAW in complementary directions. Another study investigates strategic benefits of pPoW withholding [19].

Authors of [20] show that selfish mining is an attack on the difficulty adjustment algorithm (DAA) of Bitcoin and a temporal revenue analysis shows that selfish mining strategy is sub-optimal compared to honest mining absent any DAA. The analysis of [11] discovers that even though BWH and FAW attacks are also sub-optimal, PAW strategy dominates the honest mining strategy for some parameters. Further, the authors prove that the honest miners outside the target pool profit from the PAW attack, in cases even more than the adversary mounting the attack.

Block-dependent round-based schemes also invite another attack family called *pool hopping*: an adversary leaves the pool for solo mining or another pool as the shares accumulate. Rosenfeld [2] and Raulo [21] show that a miner with negligible hashpower can increase its profit by 28% under PROP by doing so, and argue qualitatively that score-based schemes such as EXP reduce, but do not remove it. Rosenfeld also shows that PPLNS is hopping-proof as long as the difficulty and the block reward are constant. On the empirical side, [22], [23] propose methods to detect hoppers, find that hoppers earn more than static miners, including under score-based schemes, and [24] shows that payout schemes and pool fees drive the migration of miners across pools. [19], [25]–[28] study the incentive to delay the report of shares or blocks within a pool.

All works mentioned above on BWH [5]–[7], [9]–[18] consider block-dependent payout schemes. However, many contemporary mining pools employ share-based schemes. The analyses presented for block-dependent payout schemes rely on the mainstream zero-sum game framework (revenue ratio maximization) and do not extend to share-based schemes. In share-based schemes, pool operator reserves become part of the strategic interaction and invalidate the zero-sum assumption. In his analysis of the sabotage attack [2], Rosenfeld claims that BWH against PPS does not increase the attacker's rewards but instead aims to harm the pool operator. Similarly, works such as [25], [29] also consider PPS schemes and assert that delaying the reporting of a fPoW yields no profit, concluding that the scheme is incentive compatible. These conclusions are drawn under a fixed block difficulty. A related work [30] considers a solo adversary who bribes the members of a PPS pool to withhold their fPoWs, which requires a smart contract on a third-party platform, members willing to accept bribes and coordination among them.

Scheme	Withholding	Hopping (PAH)	Gain before DA	Overall
PPS	High (AoA)	Immune	None (AoA)	Vulnerable, loss on operator
PPLNS	Low (T-PAW)	Immune	Negligible (T-PAW)	Most robust
PROP	Low (T-PAW)	Very high	Immediate (PAH)	Vulnerable to PAH
EXP	Moderate (T-PAW), grows with decay rate	Moderate, decreases with decay rate	Immediate (PAH, T-PAW)	No decay rate protects against both
GRP	Overstates PROP, understates EXP	–	–	Not reliable/used in practice

TABLE II: Summary of the vulnerability of pool payout schemes to withholding and hopping attacks.

A. Gaps in the Literature

The works above leave four gaps. First, the analyses of power adjusting withholding attacks [10]–[12] adopt a global-reset version of PROP (GRP), under which a released fPoW pays the shares submitted since the last block of the whole network, which is rarely, if ever, used in practice, and further approximate how the reward of a released fPoW is shared within the pool. Since the gain of these attacks stems from changing the power allocation within a round, it is not clear whether their conclusions carry over to the round-based PROP, PPLNS and score-based schemes that pools deploy, either in the long run or before the DA. Second, the incentive compatibility of share-based schemes such as PPS has only been studied under a fixed difficulty [2], [25], [29], whereas the DA is the very mechanism that makes withholding profitable [4], [20]. Third, the pool hopping analyses [2], [21] assume a miner with negligible hashpower that either mines fully for a pool or solo, and the vulnerability of EXP to hopping has not been analyzed in closed form, although hopping under score-based schemes is observed in practice [22]. Fourth, withholding, hopping and share-based schemes have been studied separately, thus it is not known which attack is the most effective against each payout scheme, and which scheme is the most robust overall.

B. Our Contributions

In this paper, we address these gaps with the following contributions (see also Tables I and II):

- We propose *Temporary Power Adjusting Withholding* (T-PAW), where the adversary withholds a fPoW for at most a deadline T, which contains FAW and PAW as special cases. We develop an exact analysis for the revenue ratios of T-PAW by renewal arguments for PPLNS and through the stationary carry process of the open round for PROP and EXP, without approximating how the reward of a released fPoW is shared.
- The deadline of T-PAW improves on PAW (especially under EXP: by up to more than an order of magnitude). We further characterize the short-term profitability: T-PAW profits before the DA under EXP, whereas under PROP and PPLNS a pre-adjustment gain is possible albeit small, and it is the deadline that makes it larger than that of PAW. We show that GRP follows as the special case of PROP with zero carry and is not a reliable proxy for the schemes used in practice: it overstates the gains under PROP by up to more than an order of magnitude and understates it under EXP.
- We show that PPS is not incentive compatible once the DA is taken into account. The optimal BWH strategy is the *All-out Attack* (AoA), where the adversary allocates all its hashpower to the target pool and withholds every fPoW. After the first DA, AoA raises the revenue per epoch of an adversary with a fraction α of the hashpower from α to $\frac{\alpha}{1-\alpha}$, which coincides with the upper bound of optimal selfish mining under perfect network influence, while the pool operator bears all losses. Further, unlike the bribery attack of [30], AoA is mounted by a single adversary without any bribes or coordination.
- We introduce *Power Adjusting Hopping* (PAH), where an adversary of any size adjusts its power allocation to the pool according to the blockwork age of the open round without withholding any block. We derive its revenue ratios in closed form under PROP and EXP, recover the classical 28% gain of a negligible miner under PROP as the adversarial hashpower vanishes, and show that, since PAH wastes no hashpower, its gain does not rely on the DA, i.e., its profit lag is zero.
- We compare all attacks across all schemes and show that each scheme is vulnerable to a different attack: PROP is dominated by PAH, EXP cannot be tuned against both PAH and T-PAW by its decay rate, PPS is vulnerable to AoA, and PPLNS, which is immune to PAH, yields the smallest withholding gain among all

schemes, including AoA under PPS. We verify all analytical results for block-dependent schemes with Monte Carlo simulations.

C. Organization

The rest of the paper is organized as follows. Section II presents the system model and the metrics. Section III describes the payout schemes we consider. Section IV revisits the classical BWH attack under block-dependent schemes. Section V introduces T-PAW and derives its reward fractions under every block-dependent scheme, and Section VI analyzes BWH under PPS and derives AoA. Section VII analyzes PAH. Section VIII compares all attacks across all schemes, and Section IX concludes the paper. The proofs and the Monte Carlo verifications are provided in the appendices.

II. System Model

We study a Nakamoto-style blockchain secured by PoW, where miners extend the longest chain. The hashpower is spread over many miners, each holding a negligible share, and the whole network finds blocks as a Poisson process of rate λ_1, e.g., $\lambda_1 = 1/600$ per second in Bitcoin. Hence, any party that controls a fraction x of the hashpower finds blocks as an independent Poisson process of rate $x\lambda_1$.

Participants: A coalition of colluding miners that controls a fraction α of the hashpower acts as a single entity, which we call the adversary. A single mining pool, whose members are all honest and which controls a fraction β of the hashpower, is the target of the adversary, and the remaining fraction $\xi = 1 - \alpha - \beta$ belongs to honest miners outside the target pool. Honest miners always build on the tip of the longest chain they know and publish every block they find without delay. We assume $\alpha + \beta < 0.5$. The adversary may divide its hashpower between the target pool and mining on its own; we refer to the latter as *solo* mining, during which the adversary follows the protocol and publishes each block it finds immediately.

Pool mining: The operator of the target pool distributes work to its members, who report two kinds of *shares*. fPoWs are valid blocks, while pPoWs satisfy a lower difficulty and serve as evidence of the work done. If d_p and d_f denote the pPoW and fPoW difficulties, each share is also a fPoW with probability $d_p/d_f = d_b$, independently of other shares. We quantify hashing in units of *blockwork*, the amount of hashing that yields one fPoW in expectation. A party with hash fraction x performs the blockwork $x\lambda_1 t$ during a time t, the number of fPoWs it finds is Poisson with this mean, and each share carries the blockwork d_b. The operator divides the block reward among the members according to a payout scheme.

Forks: In line with the literature on incentive attacks [3], [9], we assume that the adversary observes every block as soon as it is found, and we neglect the accidental forks among honest blocks caused by propagation delays. Forks therefore arise only from the adversary's actions. When the adversary publishes a withheld fPoW in response to a block of an honest miner outside the pool, a γ fraction of the outside honest miners continue on the adversarial block and the remaining $1 - \gamma$ fraction on the honest one. The members of the target pool follow the choice of their operator. We write $\mathbb{1}_{P_R} = 1$ if the operator is rational, i.e., it builds on the fPoW released into its own pool and $\mathbb{1}_{P_R} = 0$ otherwise.

Difficulty adjustment: We assume that the blockchain adopts a DAA similar to that of Bitcoin. Let us call the blocks that become part of the longest chain in the long run as *canonical blocks*. The blockchain readjusts the mining difficulty after every D_0 canonical blocks are mined to maintain a steady block production rate of $\frac{D_0}{\tau_0}$ where the times between adjustments are called *epochs*. More specifically, given the current mining difficulty of $d_{f,c}$ in an epoch and the epoch lasts τ_c time to produce D_0 blocks, the blockchain readjusts the difficulty at the beginning of the next epoch as $d_{f,n} = d_{f,c}\frac{\tau_0}{\tau_c}$.[1] The block redundancy ratio δ^S of a strategy S is the long-run expected number of all blocks found, including orphaned and discarded ones, per canonical block.

A. Definitions

Here, we list a series of notations used frequently throughout the paper in Table III and define the following useful metrics.

[1]For the sake of convenience, we assume that a higher difficulty means fewer blocks mined. Further, even though many DAAs such as that of Bitcoin restrict the change to be between $d_{f,n} \in [d_{f,c}/4, d_{f,c} \cdot 4]$, our system model guarantees $d_{f,n} \in [d_{f,c}/2, d_{f,c} \cdot 2]$.

Parameter	Description/Value
λ_1	total block generation rate
α	adversarial hash fraction
β	target pool hash fraction
γ	fraction of outside honest miners on the adversarial fork
$\mathbb{1}_{P_R}$	1 if the pool operator is rational
T	withholding deadline (T-PAW)
t^*	hopping threshold (PAH)
W	PPLNS blockwork window
λ_d	EXP decay rate
p_i $(i=1,2,3,y,o)$	fraction of α at pool mining
$q_i,\ a_i,\ a_i'$	$1-p_i,\ \beta+\alpha p_i,\ \frac{\alpha p_i}{a_i}$
ξ	$1-\alpha-\beta$
$\pi_A,\ \pi_P,\ \pi_R$	$\frac{\alpha q_2}{1-\alpha p_2},\ \frac{\beta}{1-\alpha p_2},\ \frac{\xi}{1-\alpha p_2}$
c	$\gamma\xi+\alpha q_3+a_3\mathbb{1}_{P_R}$
$\lambda_2,\ \mu_2$	$(1-\alpha p_2)\lambda_1,\ \frac{\lambda_2}{\lambda_1 a_2}$
$\kappa_o,\ \kappa_1$	$\frac{\lambda_1 a_o}{\lambda_d},\ \frac{\lambda_1}{\lambda_d}$
$\theta,\ \hat{T},\ \chi_T$	$e^{-\lambda_2 T},\ \lambda_1 a_2 T,\ e^{-\lambda_d T}$
$y^*,\ e^*,\ \chi^*$	$\lambda_1 a_y t^*,\ e^{-y^*},\ e^{-\lambda_d t^*}$
$E_1(t)$	$\int_t^\infty \frac{e^{-x}}{x}dx$

TABLE III: Frequently used notations.

Definition 1 *The **Revenue Ratio** ρ_x^S of an entity x is the long-run fraction of the total rewards issued in the system that x receives when the adversary adopts the strategy S. The **Relative Extra Revenue (RER)** [10] of x under S_1 with respect to S_2 is*

$$\mathrm{RER}_x^{S_1,S_2}=\frac{\rho_x^{S_1}-\rho_x^{S_2}}{\rho_x^{S_2}}. \tag{1}$$

Since $\rho_A^H=\alpha$ under honest mining strategy H, we write $\mathrm{RER}=\mathrm{RER}_A^{S,H}=\frac{\rho_A}{\alpha}-1$ for the adversary A when S is clear from the context.

Definition 2 ***Revenue Change*** *[11], [31] of an entity x at time t when the adversary adopts the strategy S compared to when the adversary adopts the honest strategy H is*

$$\Delta_x^{S,H}(t)=\begin{cases}(\rho_x^S-\rho_x^H\delta^S)\frac{t}{\delta^S\tau_0}, & t\le\delta^S\tau_0,\\ \rho_x^S-\rho_x^H\delta^S+(\rho_x^S-\rho_x^H)\frac{t-\delta^S\tau_0}{\tau_0}, & t>\delta^S\tau_0,\end{cases} \tag{2}$$

*where δ^S represents the block redundancy ratio when the adversary adopts the strategy S.[2] Similarly, **Relative Revenue Change** is defined as $\overline{\Delta}_x^{S,H}(t)=\frac{\Delta_x^{S,H}(t)}{\rho_x^H}$.*

Definition 3 ***Profit Lag*** *[11], [31] of an entity x when the adversary adopts the strategy S is defined as*

$$\mathcal{T}_x^{S,H}=\inf\{s:\Delta_x^{S,H}(t)>0,\forall t>s\}. \tag{3}$$

Remark 1 *The revenue ratio and the expression of the revenue change above apply to block-dependent schemes, where all rewards stem from canonical blocks. Under share-based schemes, the pool operator pays the shares from its reserves, hence we derive the revenue changes of all entities, including the operator, directly in Section VI,*

[2]Throughout the paper, we consider a reference time $t=0$ as the starting time of a new epoch and the adversary mounts a strategy S at $t=0$, before which everyone including the adversary was following the honest protocol. For the notion of revenue change, we assume that the total rewards issued in an epoch is scaled to 1 unit, i.e., each canonical block carries the reward $1/D_0$.

while the relative revenue change and the profit lag are defined as above. The revenue change of the adversary A at $t_1 = \delta^S \tau_0$ tells us how much its revenue changes at the end of the first epoch compared to the case if it adopted the honest strategy H. Note that $\Delta_x^{H,H}(t) = 0, \forall(t, x)$ since $\delta^H = 1$. If $\Delta_A^{S,H}(t_1) > 0$, the profit lag of the adversary is zero, i.e., under strategy S the adversary gets more revenue than honest mining even before the first DA. We refer the reader to [11], [31] for a more detailed explanation to understand the intuition behind the metrics defined above.

III. Pool Reward Payout Schemes

We focus on mainstream mining pool payout schemes that can be divided into two distinct categories, namely share-based schemes and block-dependent schemes. In block-dependent schemes, such as PROP, PPLNS and EXP, miners are paid only when the pool actually discovers a valid block. In share-based schemes such as PPS and FPPS, each valid share submitted to pool operator gets an immediate payment from the pool operators that is valued at a fraction of the coinbase reward. In this paper, we study PPS, PROP, PPLNS and EXP: these schemes cover the main design ideas, and together they span the space of mainstream payout schemes well.

A. Share-based Schemes: Pay-Per-Share (PPS) and variants

PPS is a share-based scheme in which each valid share receives an immediate payment from the pool operator proportional to its expected statistical value of the coinbase reward. More specifically, since each share is also a fPoW with probability $d_b \ll 1$, on average it takes $1/d_b$ shares before the pool discovers one block and receives one block reward, denoted as R_b. Consequently, each submitted share, regardless of whether it is a pPoW or fPoW solution, earns an immediate payment equal to

$$R_s = R_b \times \frac{\textit{share difficulty}}{\textit{block difficulty}} = R_b d_b. \tag{4}$$

Note that PPS has a variant, FPPS, where each share receives an immediate average proportional share of transaction fees, in addition to its statistical coinbase reward share. Although our theoretical analysis of PPS in Section VI considers only coinbase rewards, the results generalize to FPPS by treating the effective block reward as the sum of two components: the coinbase reward and the transaction fees.

Since each share receives an immediate fixed payment, miners enjoy a stable cash flow, while all risk associated with the variance of block mining times is borne by the pool operator. Consequently, during unlucky periods when the pool fails to discover a block, the operator incurs losses; conversely, during lucky periods, the operator realizes additional profits. As such, PPS operators require substantial reserves and strong capital backing. Many modern pool operators implement this payout scheme in practice. Indeed, PPS/FPPS remains widely deployed in Bitcoin mining. For example, Foundry USA (approximately 25–28% of the network hashrate), AntPool (approximately 19–21%), F2Pool (approximately 12–14%), and ViaBTC (approximately 10–11%) collectively account for a substantial fraction of Bitcoin's total hashrate [32]. Among these, Foundry USA employs FPPS [33], AntPool supports FPPS, PPS and PPLNS [34], ViaBTC offers PPS+ and PPLNS [35], while F2Pool supports FPPS/PPS variants [36].

B. Block-Dependent Payout Schemes

A block-dependent payout scheme is one where the miners are paid only when the pool actually discovers a valid block. Although shares do not get direct payments, the final payment a miner receives when a block is discovered is a function of the share contributions. Since miners only get payments when the pool finds blocks, the cashflow of miners is less stable compared to share-based schemes such as PPS, where each valid share submitted to pool operator gets an immediate payment from the pool operators that is valued at a fraction of the coinbase reward. Hence, in block-dependent schemes the risk associated with the variance of block mining process is borne by the pool miners collectively. These schemes are attractive because pools need smaller reserves, operator risk is much lower and as a result miners face lower fees.

1) Pay-Per-Last-N-Shares (W-PPLNS):
PPLNS is a sliding-window payout scheme: when the pool finds a block, the block reward R_b is split proportionally among the last N shares submitted to the pool, i.e., a miner who contributed n of these N shares receives $R_b\frac{n}{N}$. We parametrize the window by its blockwork $W \triangleq Nd_b$ and refer to the scheme as W-PPLNS since N and W are in one-to-one correspondence for given difficulties. In real-world PPLNS implementations, N typically covers a small number of blockworks [22], e.g., $W = 2$ [37], [38].

2) Proportional (PROP):
PROP is a simple round-based payout scheme where a block discovery triggers reward split proportionally among all shares submitted during that round. In this context, a round refers to the time interval between two consecutive pool blocks that become canonical. In PROP, a miner who contributes m_s shares out of the m_S total shares the pool receives during the round, receives $R_b\frac{m_s}{m_S}$ rewards at the end of the round.

3) Score-based: Exponential-decay (EXP):
Score-based schemes are mining-pool reward systems where each share gets a time-dependent weight that gradually decreases as it gets older. In this paper, we specifically focus on historically deployed slush-style round-based modeling with exponential-decay, where the shares within a round are weighted with an exponential function $w(t) = e^{-\lambda_d t}$, where λ_d is the decay rate, which we express relative to the total block generation rate as $\frac{\lambda_1}{\lambda_d} = \kappa_1$ (e.g., $\kappa_1 = 1/2$ in the original slush method [2], [39]). More specifically, when a round ends, each share i that is mined t_i time ago within that round by a miner j gets a weight $w(t_i) = e^{-\lambda_d t_i}$ and miner j receives $\frac{\sum_{i\in j} w(t_i)}{\sum_k w(t_k)}$ reward at the end of the round.

4) Global-reset-PROP (GRP):
Recent analyses of advanced block withholding attacks [10]–[12] implicitly adopt an unconventional version of PROP scheme that is rarely, if ever, used in practice, which we call GRP. Under GRP, rounds start and end whenever any block becomes known to the pool operator, whether it is mined by the pool or by an entity outside the pool. Since only the shares submitted since the last block of the network are paid, pool miners bear considerably more risk than under PROP. On the other hand, GRP is more appealing from an academic perspective, as it is considerably cleaner to analyze. As we show later, this simplification is not innocuous: GRP gives more weight to recent shares and therefore overstates the gain of power adjusting attacks compared to the round-based PROP.

IV. Classical Block Withholding Attacks Under Block-Dependent Payout Schemes

Let us briefly revisit classical BWH attack under a block-dependent payout scheme, in which pool miners are paid according to their share of contributions when the pool discovers a block. Suppose the adversary joins the target pool using p fraction of its hashpower, submitting pPoWs it encounters but never submitting any fPoW. The remaining $(1-p)$ fraction of its hashpower is used to mine individually and honestly (referred to as "solo" mining). Under such an attack with a block-dependent payout scheme, we can directly apply the zero-sum game principle. The revenue ratios for the adversary (ρ_A), honest target pool members (ρ_P), and the rest of the honest miners (ρ_R) are given by (see for example [6])

$$\rho_A = \frac{\alpha(1-p) + \beta r}{1-\alpha p}, \tag{5}$$

$$\rho_P = \frac{\beta(1-r)}{1-\alpha p}, \tag{6}$$

$$\rho_R = \frac{\xi}{1-\alpha p}. \tag{7}$$

Here, r denotes the average adversarial share of rewards per block discovered by a pool member and is given by

$$r = \frac{\alpha p}{\beta + \alpha p} \tag{8}$$

for all block-dependent schemes considered here, since the adversarial power allocation towards the target pool is constant. Further, the block redundancy ratio under BWH is

$$\delta^{(p)} = \frac{1}{1-\alpha p}. \tag{9}$$

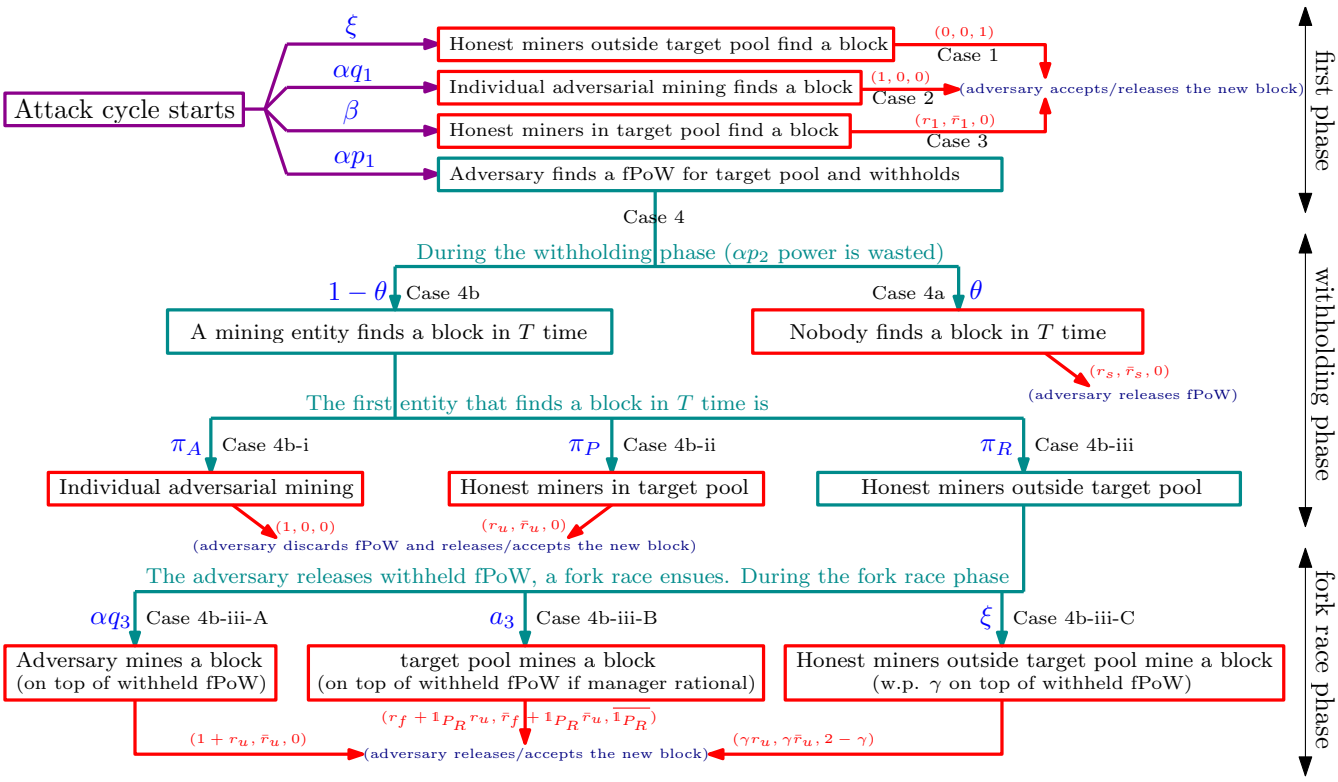

Fig. 1: Evolution of a T-PAW attack cycle.

BWH has two natural extensions, FAW and PAW. In **FAW**, the adversary releases a withheld fPoW when any miner outside the target pool mines a block. This triggers a fork race, which can increase adversarial rewards compared to classical BWH [9]. **PAW** generalizes FAW by allowing the adversary to adjust p to p_2 whenever it encounters and withholds a fPoW, thereby potentially increasing its profit. Notably, setting $p_2 = p$ reduces PAW to FAW [10]. An adversary can choose p (and p_2 in PAW) to maximize its revenue ratio, thereby increasing its profit relative to honest mining.

V. Advanced Block Withholding Attacks Under Block-Dependent Payout Schemes

In this section, we focus on the state-of-the-art variants of block withholding attacks where the adversary can adjust its power allocation towards the target pool to maximize its gains. Our goal is to present the relevant formulas associated with these attacks, including revenue ratios and block redundancy ratios, under the implicit assumption of a block-dependent payout scheme. The revenue ratios depend on the payout scheme only through the adversarial reward fractions of the pool blocks. Then, we provide novel and exact formulas of the reward fractions for all four block-dependent schemes listed in Section III-B.

To do so, we first propose an improvement to the original PAW attack: *Temporary PAW* (T-PAW), which is essentially the same strategy as PAW however there is a deadline on block withholding, i.e., the adversary releases the withheld block after at most T-time. We show that PAW attack corresponds to $T \to \infty$ and is not optimal in general. Hence, T-PAW is the third natural extension of BWH building upon PAW [10] which builds upon FAW [9]. On a high level, the extensions enable us to state that FAW $\subset$ PAW $\subset$ T-PAW in terms of the attack space, thus we directly present the T-PAW scheme here. T-PAW also contains the lie-in-wait attack of [2], in which the adversary temporarily withholds a fPoW while increasing its power allocation to the pool. Notably, lie-in-wait was proposed for PPLNS pools and analyzed only approximately, i.e., for a small withholding duration and without fork races.

A. Temporary Power Adjusting Withholding (T-PAW)

1) Modeling Issues in Previous Literature:
The model of FAW [9] and PAW [10] assume some simplifications in the fork races pertaining to the attack whereas

a more rigorous analysis of the attacks is provided in [11]. The simplified model assumes that each attack cycle contributes one block to the longest chain growth and is independent from other cycles whereas the latter model considers the fork races explicitly until the attack cycles fully end using γ. Since T-PAW attack can involve multiple block growth in an attack cycle, in this paper, we use the rigorous modeling of [11].

However, although [11] builds a more rigorous analysis compared to [10], both models use the GRP payout scheme, as it is considerably easier to analyze, but it overstates the gain of power adjusting attacks compared to actual PROP used in practice. Despite GRP being considerably easier to analyze, both models further resort to an approximation regarding the rewards of the fPoWs shared between pool members and the adversary (see [11, Appendix B, Footnote 13]). The approximation error grows large especially for T-PAW attacks considered in this paper. Hence, in this paper, we introduce a general framework to analyze the T-PAW attack where we provide an explicit and rigorous analysis regarding the rewards of the fPoWs, which enables us to derive exact results rather than approximations for all four block-dependent schemes listed in Section III-B.

2) T-PAW Attack Description:

Here, we describe how a T-PAW attack cycle works which contains FAW and PAW attacks as special cases. To avoid repetition, when we explain the attack cycle in the next paragraph, we advise the reader to follow Fig. 1, which contains all the relevant cases and phases, as well as the corresponding probabilities and rewards. Note that, in Fig. 1 an attack cycle starts with the purple box and ends with any of the events listed in the red box. The arrows represent the transition to subcases and blue symbols near the arrows show the probabilities. Each red box ends the cycle and points to the rewards each entity gets from the cycle as a 3-tuple $(\cdot, \cdot, \cdot)$ where the fields are the rewards of the adversary, the honest target pool miners and the remaining honest miners respectively.

Let us assume everyone agrees on the last block mined in the network and a new attack cycle starts, where the adversary mines solo with q_1 fraction of its hashpower and contributes to the pool with the remaining p_1 fraction. This phase is called *first phase* and the first miner to mine a block is:

Case 1: Honest miners outside the target pool. The adversary accepts the block and the cycle ends.

Case 2: Solo adversary. The adversary releases the block and the cycle ends.

Case 3: Honest miners of the target pool. The adversary accepts the block and the cycle ends.

Case 4: The adversarial power in the target pool. The adversary withholds the fPoW and readjusts its power contributing to the target pool as $p_1 \rightarrow p_2$. This phase is called *withholding phase* where the adversary continues mining; however, if it encounters another fPoW with its pool power p_2, it discards the old fPoW and keeps the new one. If:

Case 4a: No entity other than adversarial pool mining power finds a block for T time duration, the adversary releases the withheld fPoW and attack cycle ends.

Case 4b: The first entity that finds a block in T time is:

Case 4b-i: Solo adversary. The adversary discards its withheld fPoW and releases the solo block and the cycle ends.

Case 4b-ii: Honest miners of the target pool. The adversary discards its withheld fPoW and accepts the new pool block and the cycle ends.

Case 4b-iii: Honest miners outside the target pool. The adversary releases its withheld fPoW and a fork race ensues. During the *fork race phase*, the adversary readjusts its power contributing to the target pool as $p_2 \rightarrow p_3$ and tries to mine on top of the fPoW it recently released. The first entity that finds a block during the *fork race phase* is:

Case 4b-iii-A: Solo adversary. The adversary releases the block and the cycle ends.

Case 4b-iii-B: Honest miners of the target pool or the adversarial pool mining. Note that if the target pool operator is rational, this pool block is mined on top of the recently released fPoW by the adversary. In either case, the adversary accepts/releases the block and the cycle ends.

Case 4b-iii-C: Honest miners outside the target pool. Note that γ fraction of these miners prefer the recently released fPoW by the adversary. In either case, the adversary accepts/releases the block and the cycle ends.

When the cycle ends, the adversary sets its power allocation towards the target pool as p_1 and a new attack cycle starts.

3) Relation to Existing Strategies:

First, note that when $T = 0$, the T-PAW reduces to honest mining strategy irrespective of p_1, p_2, p_3 values. On the

other hand, when $T \to \infty$, T-PAW reduces to PAW described in [11] with an additional flexibility for the adversary which can readjust its power allocation towards the target pool as $p_2 \to p_3$ during the *fork race phase* (the authors assume $p_3 = 0$ in [11]). This flexibility does not necessarily bring major improvements; however, it makes FAW, where the adversary never changes its allocation, an exact special case of T-PAW with $p_1 = p_2 = p_3 = p$ and $T \to \infty$, under which all adversarial reward fractions reduce to r.

Our analysis suggests that for many parameters, especially when α, β, γ are small, $T \to \infty$ is not optimal and there is a room for improvement compared to PAW. Such an argument might be counterintuitive when one takes the memorylessness property of exponential arrivals into account since the first arrival time in Case 4b does not depend on how long the adversary already withheld the fPoW. However, the reward of the adversary from the fPoW is not memoryless, which affects the final revenue ratio. For example, under PROP, withholding the fPoW and increasing p_1 to p_2 increases the adversarial reward fraction for a round, but as more time passes the reward fraction saturates and the adversary only risks the fPoW becoming canonical as an honest miner outside can potentially create a block which will start a fork race.

It is claimed in [11] that for many parameters, the PAW attack can be superior to honest mining even when there is no DAA in the protocol.[3] On the other hand, without DAA, withholding strategies such as selfish mining result in less revenue than honest mining [20]. An inspection of such a situation in [11, Example 1] with $T = \infty$ suggests that the phenomenon occurs especially when p_2 is large and β is small and γ is large. In such a situation, the adversary significantly increases p_1 to p_2, since if the fork race resolves in its favor, it will get more revenues from fPoW than it would if it were to act honestly and publish immediately.

Notice, when an attack cycle starts and the adversary encounters fPoW, it withholds and readjusts its power allocation to the pool until some other entity finds a block in the regular PAW attack ($T = \infty$). **There are two shortcomings of this strategy even if p_2 is large and β is small**: 1) If it takes a long while until someone else finds a block, the total adversarial fractional contribution to the pool gets saturated. 2) If γ is not large, the adversary is risking all the contributions to fPoW by waiting more since the fork race is less likely to result in its favor. **Thus, T-PAW proposes that the adversary withholds the fPoW at most for a finite time T that is to be optimized in addition to p_1, p_2, (and p_3) which in turn brings improvements.**

4) Analysis of T-PAW Quantities:

It is fairly straightforward to verify each of the probabilities listed in Fig. 1. Regarding the 3-tuple rewards, note that a pool block released at the first phase is distributed between the adversary and the honest pool members as $(r_1, \bar{r}_1)$. In Case 4a, the reward fractions of the released fPoW are $(r_s, \bar{r}_s)$. On the other hand, in Case 4b, the honest pool block of Case 4b-ii and the released fPoW of Case 4b-iii, if it ends up being canonical, both have the reward fractions $(r_u, \bar{r}_u)$. Note that, the pool may mine an extra block in Case 4b-iii-B (during the fork race), which has reward fractions $(r_f, \bar{r}_f)$. Using this parametrization and $\bar{r}_x = 1 - r_x$ with $x \in \{1, s, u, f\}$ it is straightforward to verify the 3-tuple rewards of each case as shown in Fig. 1. It remains to combine the total rewards of each case with the corresponding probability and then explicitly derive the reward fractions r_1, r_s, r_u, r_f.

During an attack cycle, let B_A, B_P and B_R denote the reward obtained by the adversary, honest target pool miners and honest miners outside the target pool, respectively. Let B_C and B_O denote the total numbers of canonical blocks and overall blocks generated during an attack cycle, respectively. Combining the rewards and probabilities of each case in Fig. 1, we have

$$\rho_x = \frac{\mathbb{E}[B_x]}{\mathbb{E}[B_C]}, \tag{10}$$

where $x \in \{A, P, R\}$ with

$$\mathbb{E}[B_C] = 1 + \alpha p_1 (1 - \theta) \pi_R, \tag{11}$$

$$\mathbb{E}[B_A] = \alpha q_1 + \beta r_1 + \alpha p_1 \Big(\theta r_s + \big(1 - \theta\big) \Big(\pi_A + \pi_P r_u + \pi_R \big(c r_u + a_3 r_f + \alpha q_3 \big) \Big) \Big), \tag{12}$$

$$\mathbb{E}[B_P] = \beta \bar{r}_1 + \alpha p_1 \Big(\theta \bar{r}_s + \big(1 - \theta\big) \Big(\pi_P \bar{r}_u + \pi_R \big(c \bar{r}_u + a_3 \bar{r}_f \big) \Big) \Big), \tag{13}$$

[3]The authors of [11] derive this result for PAW under GRP. In Section VIII, we confirm that T-PAW, which generalizes PAW, can profit before the DA under the schemes used in practice as well, although this gain is negligible under PROP and PPLNS.

$$\mathbb{E}[B_R] = \xi\bigg(1 + \big(1-\theta\big)\frac{\alpha p_1}{1-\alpha p_2}\big(\xi(2-\gamma) + a_3\overline{\mathbb{1}_{P_R}}\big)\bigg). \tag{14}$$

Here, we note that the reward fractions of a pool block may depend on blockwork submitted in earlier attack cycles, hence $\mathbb{E}[B_A]$ and $\mathbb{E}[B_P]$ are understood as long-run (stationary) expectations per attack cycle, and r_1, r_s, r_u, r_f as the corresponding stationary adversarial reward fractions. Further, since the events that end an attack cycle are races between independent exponential clocks, their probabilities and the number of blocks they create do not depend on how the pool shares rewards. Hence, $\mathbb{E}[B_C]$, $\mathbb{E}[B_R]$, $\mathbb{E}[B_O]$, and therefore ρ_R and δ, are the same for every block-dependent payout scheme and do not depend on the adversarial reward fractions r_1, r_s, r_u, r_f. On the other hand, $\mathbb{E}[B_A]$, $\mathbb{E}[B_P]$ keep their form but the adversarial reward fractions r_1, r_s, r_u, r_f depend on the scheme, i.e., the payout scheme only moves reward between the adversary and the honest pool members. The exact values of these adversarial reward fractions are provided later for each scheme. It is straightforward to verify that $\rho_R \geq \xi$ and $\rho_A + \rho_P + \rho_R = 1$. Further, the block redundancy ratio is

$$\delta^{\mathrm{T-PAW}} = \frac{\mathbb{E}[B_O]}{\mathbb{E}[B_C]}, \tag{15}$$

where

$$\mathbb{E}[B_O] = \mathbb{E}[B_C] + \alpha p_1\bigg(\theta\alpha p_2\lambda_1 T + \big(1-\theta\big)\Big(1 + \alpha p_2\lambda_1 f_{\lambda_2}(T)\Big)\bigg), \tag{16}$$

with $f_\lambda(t) = \frac{1}{\lambda} - \frac{te^{-\lambda t}}{1-e^{-\lambda t}}$. The goal of the adversary is to optimize an objective function, such as the revenue ratio ρ_A, with respect to p_1, p_2, p_3 and T. In general, objective functions such as ρ_A are neither concave nor convex.

B. Reward Fractions Under FAW, PAW, T-PAW

For block-dependent schemes, the adversarial reward fraction of a pool block is the adversarial fraction of the shares that the scheme pays for that block, i.e., the shares in the last W blockworks under PPLNS, the shares of the round under PROP, the decay-weighted shares of the round under EXP, and the shares since the last block of the network under GRP. Since the adversary submits shares with different fractions a_1', a_2', a_3' in different phases, reward fractions depend on which past phases the scheme pays and with what weight.

First, note that r_s does not appear in the final expression for FAW and PAW since $T \to \infty$ implies $\theta = 0$. Further, the analysis does not depend on the particular values of T and λ_1 but only on the product $\lambda_1 T$. In other words, the adversary can first pick $\lambda_1 = 1$, find the optimal T and rescale it with the actual block production rate λ_1. Moreover, the adversarial reward fractions r_s and r_u are affected by the adversarial power adjustments towards the target pool during the withholding phase and deviate from r_1 accordingly. This deviation is more pronounced in schemes where recent shares have higher weight, such as EXP or GRP, which makes them more susceptible to PAW/T-PAW as we will see later.

1) Reward Fractions in FAW:

In FAW, the adversary does not alter its power allocation devoted to pool mining, i.e., $p_i = p$ for $i = 1, 2, 3$ and $T \to \infty$. It is straightforward to show that the adversarial reward fraction for FAW under all four block-dependent payout schemes listed in Section III-B remains equal to r given in (8); that is,

$$r_1^{FAW} = r_u^{FAW} = r_f^{FAW} = r. \tag{17}$$

2) Reward Fractions for W-PPLNS:

Let S be the blockwork of the target pool in an attack cycle. An analysis of W-PPLNS scheme is provided in Appendix A using renewal equations and the following result is obtained.

Theorem 1 *With $W \gg \mathbb{E}[S]$, up to vanishing terms,*

$$r_1 = \bar{x} + \frac{1}{W}\Big[\big(a_1' - \bar{x}\big)\, a_1 + H_0\Big], \tag{18}$$

$$r_s = \bar{x} + \frac{1}{W}\Big[\big(a_1' - \bar{x}\big)\, a_1 + \big(a_2' - \bar{x}\big)\,\hat{T} + H_0\Big], \tag{19}$$

$$r_u = \bar{x} + \frac{1}{W}\Big[\big(a_1' - \bar{x}\big)\, a_1 + \big(a_2' - \bar{x}\big)\,\nu_1 + H_0\Big], \tag{20}$$

$$r_f = \bar{x} + \frac{1}{W}\Big[(a_1' - \bar{x})\, a_1 + (a_2' - \bar{x})\, \nu_1 + (a_3' - \bar{x})\, a_3 + H_0\Big], \tag{21}$$

with $\bar{x}$ in (74)*, H_0 in* (77) *and ν_1 in* (64)*. Here, $\bar{x}$ is the long-run adversarial fraction of the pool blockwork, and the $1/W$ terms correct it for the blockwork of the current attack cycle, which lies at the end of the window and has the adversarial fraction a_i' in phase i. The exact expressions are given in Theorem 7.*

3) Reward Fractions for PROP:
An analysis of PROP scheme is provided in Appendix B using a carry process that models how the blockwork of attack cycles that produce no canonical pool block is carried over to the next cycles. With the help of the Laplace transform of the carry process, the following result is obtained.

Theorem 2 *The expected adversarial reward fractions under round-based PROP are, for $x \in \{1, s, u, m\}$,*

$$r_x = a_1' + (a_2' - a_1')\, \omega_2^x + (a_3' - a_1')\, \omega_3^x, \tag{22}$$

$$r_f = \mathbb{1}_{P_R}\, a_3' + \overline{\mathbb{1}_{P_R}}\, r_m. \tag{23}$$

Here, r_m is the adversarial reward fraction of the pool block of the fork race when the pool operator is not rational, $\omega_i^x \in [0,1]$ is the expected fraction of the total blockwork of the round submitted in mode i (first phase, withholding phase and fork race phase for $i = 1, 2, 3$), both derived in Appendix B.

4) Reward Fractions for EXP :
Under EXP, the reward of a round depends on the score $\mathbf{Z} = (Z_A, Z)$ of its shares, which is carried over from earlier attack cycles as under PROP, but discounted by $e^{-\lambda_d t}$ over time. An analysis of this scheme is provided in Appendix C, and the following result is obtained.

Theorem 3 *The expected adversarial reward fractions under round-based EXP are*

$$r_x = \int_0^\infty \mathcal{L}_A^x(s)\, ds, \qquad x \in \{1, s, u, m\}, \tag{24}$$

$$r_f = \mathbb{1}_{P_R}\, a_3' + \overline{\mathbb{1}_{P_R}}\, r_m, \tag{25}$$

where $\mathcal{L}_A^x(s) = \mathbb{E}[Z_A e^{-sZ}]$ for the score of the corresponding paid round, given in Theorem 8 in terms of the unique bounded solution of the linear fixed point equation (154)*.*

Unlike PROP, the transforms are not explicit, but every operator in (154) is a one-dimensional integral, thus (154) is solved on a grid and r_x follows by one-dimensional quadrature; see Theorem 8 for the formal statement.

5) Reward Fractions for GRP:
Under GRP every block of the network starts a new round and no blockwork is carried over from earlier attack cycles. Hence, GRP is a special case of round-based PROP with zero carry, and Theorem 2 gives (see Remark 11 in Appendix B),

$$r_1 = a_1', \tag{26}$$

$$r_s = a_1' + (a_2' - a_1')\, \frac{\hat{T}}{a_1} e^{\hat{T}/a_1} E_1\Big(\frac{\hat{T}}{a_1}\Big), \tag{27}$$

$$r_u = a_1' + \frac{a_2' - a_1'}{1-\theta} \int_0^{\hat{T}} \mu_2 e^{-\mu_2 y}\, \frac{y}{a_1} e^{y/a_1} E_1\Big(\frac{y}{a_1}\Big) dy, \tag{28}$$

$$r_f = a_3'. \tag{29}$$

VI. Block Withholding Attacks Under Share-Based Payout Schemes

BWH studies in literature focus on block-dependent payout schemes in which coinbase rewards are distributed among pool members upon block discovery. In such settings, zero-sum game analysis with revenue ratio maximization is a standard approach for studying incentive compatibility, since increasing rewards for one mining entity directly reduces those of another as coinbase rewards constitute the sole asset under consideration. In contrast, in share-based schemes, pool operators maintain capital reserves to pay members for finding pPoWs even when the

pool does not consistently discover fPoWs and receive coinbase rewards. Consequently, a zero-sum game framework does not apply; therefore, we adopt an alternative analytical technique, i.e., revenue change analysis, commonly employed in the literature to study incentive attacks such as selfish mining [4], [40].

A. Optimal BWH Under PPS: All-out Attack

In this section, we analyze the profitability of BWH under the PPS scheme. We show that the optimal BWH attack in PPS differs substantially from the mainstream zero-sum game analysis (revenue ratio maximization) applied to block-dependent schemes. Specifically, under PPS, the adversary allocates all its mining power exclusively to the target pool and never releases a withheld fPoW, a strategy we term **All-out Attack** (AoA). We argue that advanced variants of BWH do not yield additional adversarial profit under PPS; however, FAW can marginally mitigate losses for the pool operator. Our results extend to the FPPS scheme by treating the effective block reward as the sum of coinbase and transaction fees.

With the AoA, target pool operators under the PPS scheme suffer complete harm: the adversary allocates all its hashpower toward the target pool but never releases any block. Consequently, the target pool effectively pays the adversary out-of-pocket for every pPoW share. Since no block is published by the adversary, difficulty drops after the first epoch. Following the DA, the adversary begins to accrue profits. We now provide a rigorous analysis substantiating these claims. Let us assume that at time $t = 0$, the adversary joins the target pool with a fraction p of its hashpower and initiates the classical BWH attack. Under the PPS scheme, we show that the optimal adversarial power allocation is $p^* = 1$, i.e., the All-out Attack.

We first assume that the mining pool maintains a fixed difficulty ratio between fPoWs and pPoWs for the adversary, i.e., d_b remains constant and any change in fPoW difficulty is accompanied by a proportional adjustment in pPoW difficulty. Under this assumption, the reward per share R_s remains constant throughout the analysis. However, if the fPoW difficulty decreases, an honest pool miner allocating the same amount of hashpower will submit shares at a higher rate. We first analyze this setting for simplicity before extending the results to a more practical setting.

In practice, mining pools typically employ *variable share difficulty* (vardiff) to maintain an approximately constant share submission rate, e.g., one share every few seconds. Vardiff is primarily an operational mechanism that reduces server overhead and improves hashrate estimation; it does not affect a miner's expected payout. If the network block difficulty decreases while the share difficulty remains fixed, each submitted share becomes more valuable because the PPS reward is proportional to the ratio between the share difficulty and the block difficulty. Conversely, if the share difficulty is adjusted proportionally with the fPoW difficulty, the reward per share remains unchanged, while miners submit more shares per time. The main results of this paper hold under both assumptions, with Theorems 4 and 5 establishing the optimal attack strategy under each assumption. Unless stated otherwise, we neglect pool fees; see Remark 3 for the case with fees.

Lemma 1 *Assume the adversary mines honestly prior to $t = 0$, and the blockchain adjusts difficulty at $t = 0$, initiating a new epoch (the first). Under the PPS scheme, if the adversary launches a classical BWH attack against a target pool using fraction p of its hashpower from $t = 0$ onward, the duration of the first epoch is*

$$t_1 = \delta^{(p)}\tau_0 = \frac{\tau_0}{1-\alpha p}. \tag{30}$$

At the end of this epoch, the adversary's revenue change is

$$\Delta_A^{BWH,H}(t_1) = -\delta^{(p)}\alpha p d_b \approx 0, \tag{31}$$

whereas the revenue changes of the honest miners in the target pool and those outside are zero, i.e.,

$$\Delta_P^{BWH,H}(t_1) = \Delta_R^{BWH,H}(t_1) = 0, \tag{32}$$

and that of the target pool operator is

$$\Delta_{P_o}^{BWH,H}(t_1) = -\delta^{(p)}\alpha p(1-d_b). \tag{33}$$

Proof: The duration of the first epoch is scaled by $\delta^{(p)}$ because a fraction αp of the total system hashpower is effectively wasted due to the adversary's withholding behavior. Note that, in the absence of attack, an entity

controlling ρ_x^H fraction of the total network hashpower would earn ρ_x^H rewards over time interval τ_0. Thus, if the adversary were mining honestly during $[0, \delta^{(p)}\tau_0]$, its reward would be $\alpha\delta^{(p)}$, that of honest miners in the target pool would be $\beta\delta^{(p)}$, and that of the remaining honest miners would be $\xi\delta^{(p)}$, all measured over time interval $\delta^{(p)}\tau_0$. Hence, if during $[0, \delta^{(p)}\tau_0]$ the adversary allocates fraction p of its hashpower honestly toward the target pool, it would earn $\delta^{(p)}\alpha p$ PPS rewards.

Under the attack, we partition the reward analysis into two components: 1) **Coinbase rewards:** The portion attributable to coinbase rewards earned by the adversary (solo), the target pool operator, and the remaining honest miners. 2) **PPS rewards:** The portion corresponding to PPS rewards transferred from the target pool operator to its members, including both honest and adversarial contributions. This decomposition allows us to isolate the economic impact of withholding behavior on each stakeholder group under the PPS scheme.

Coinbase rewards: During the first epoch (spanning $\delta^{(p)}\tau_0$), D_0 canonical blocks are generated. Note that only $(1-\alpha p)$ fraction of the total network hashpower contributes to mining these blocks, with contributions distributed as follows: The adversary (solo) mines at rate $\alpha(1-p)$. The target pool operator mines at rate β. The remaining honest miners mine at rate ξ. Thus, the total coinbase reward received by each group during this epoch is

$$C_{1,A_S} = \frac{\alpha(1-p)}{1-\alpha p} = \alpha(1-p)\delta^{(p)}, \tag{34}$$

$$C_{1,P_o} = \frac{\beta}{1-\alpha p} = \beta\delta^{(p)}, \tag{35}$$

$$C_{1,R} = \frac{\xi}{1-\alpha p} = \xi\delta^{(p)}. \tag{36}$$

PPS submission rewards: Since block difficulty remains unchanged until the end of the first epoch (which lasts $\delta^{(p)}\tau_0$), the target pool operator distributes PPS rewards to participants based on their submitted shares. The adversary, contributing fraction p of its total hashpower to the target pool during this interval, receives

$$P_{1,A_P} = \delta^{(p)}\alpha p(1-d_b) \tag{37}$$

PPS rewards over the epoch. Note that d_b fraction of the shares generated by the adversary are withheld fPoW, resulting in a small net loss for the adversary relative to an honest mining scenario. Similarly, the target pool operator distributes

$$P_{1,P} = \delta^{(p)}\beta \tag{38}$$

PPS rewards to its honest members during this epoch.

Hence, the total reward received by the adversary in the first epoch is $C_{1,A_S} + P_{1,A_P}$ which is $\delta^{(p)}\alpha p d_b$ less than its expected revenue had it mined honestly during $[0, \delta^{(p)}\tau_0]$. Therefore, the net revenue change for the adversary is $-\delta^{(p)}\alpha p d_b$. Similarly, compared to the scenario in which the adversary mines honestly throughout the first epoch, the total reward received by honest members of the target pool (as well as those honest miners outside the pool) remains unchanged. Their respective rewards are unaffected by the adversary's withholding behavior during this interval.

Note that, the target pool operator receives only coinbase rewards equal to $C_{1,P_o} = P_{1,P}$, yet it pays out a total of $P_{1,A_P} + P_{1,P}$ in PPS payments for submitted shares. In other words, during the first epoch, the target pool operator bears an additional cost, specifically, it compensates the adversary directly for every pPoW share submitted, effectively paying out-of-pocket to offset the withholding behavior. ■

Corollary 1 *Under the PPS scheme, if the adversary initiates a classical BWH against a target pool with p fraction of its power from $t = 0$ onwards, after the first epoch, the revenue change of each mining entity is given by,*

$$\Delta_A^{BWH,H}(t_1+t) = \frac{\alpha t}{\tau_0}\big(\delta^{(p)}(1-pd_b)-1\big) - \alpha\delta^{(p)}pd_b \approx \frac{\alpha t}{\tau_0}(\delta^{(p)}-1), \tag{39}$$

$$\Delta_P^{BWH,H}(t_1+t) = \frac{\beta t}{\tau_0}(\delta^{(p)}-1), \tag{40}$$

$$\Delta_R^{BWH,H}(t_1+t) = \frac{\xi t}{\tau_0}(\delta^{(p)}-1), \tag{41}$$

whereas the revenue change of the pool operator is,

$$\Delta_{P_o}^{BWH,H}(t_1+t) = -\Big(1+\frac{t}{\tau_0}\Big)\alpha\delta^{(p)}p(1-d_b). \tag{42}$$

Proof: After the first epoch, difficulty is adjusted downward (by $\delta^{(p)}$), and since the adversary continues its classical BWH attack unchanged, each subsequent epoch lasts τ_0 time units. Without the attack, the reward rate for each entity over each τ_0 interval would be: α for the adversary, β for honest miners in the target pool, ξ for all other honest miners. With the attack active during each subsequent epoch:

- The coinbase rewards received by each entity are C_{1,A_S} (solo adversary), C_{1,P_o} (target pool operator), and $C_{1,R}$ (other honest miners).
- Due to the difficulty reduction by $\delta^{(p)}$, the target pool operator pays P_{1,A_P} in PPS rewards to the adversary over each τ_0 interval, whereas prior to this adjustment, the adversary received P_{1,A_P} over a duration of $\delta^{(p)}\tau_0$.

Hence, the revenue change for the adversary at time t_1+t (i.e., after t/τ_0 subsequent epochs) is,

$$\Delta_{A}^{BWH,H}(t_1+t) = \big(C_{1,A_S}+P_{1,A_P}-\alpha\big)\frac{t}{\tau_0} + \Delta_{A}^{BWH,H}(t_1). \tag{43}$$

The revenue changes for the other entities can be derived similarly. For the target pool operator, note that over each τ_0 interval after the first epoch, it incurs a net out-of-pocket loss of P_{1,A_P}, because these are shares submitted by the adversary that never result in coinbase rewards. ■

Theorem 4 *If $\alpha > d_b$, **All-out Attack**, i.e., picking $p^* = 1$ in BWH, maximizes the adversarial revenue change after the first DA under the PPS scheme. Otherwise, the adversary should mine honestly.*

Proof: The adversary's revenue change in the first epoch is

$$-\frac{\alpha p}{1-\alpha p}\times d_b, \tag{44}$$

which represents a small loss since $d_b \ll 1$. After the first DA, the rate of change of adversarial revenue (its slope) is

$$\frac{\alpha}{\tau_0}\left(\delta^{(p)}(1-pd_b)-1\right), \tag{45}$$

which is positive if and only if

$$\delta^{(p)} > \frac{1}{1-pd_b}, \tag{46}$$

equivalent to $\alpha > d_b$. Moreover, under the condition $\alpha > d_b$, this slope is increasing in p. Therefore, to maximize revenue change after the first adjustment, the adversary should set $p = 1$, i.e., adopt the All-out Attack. If $\alpha \le d_b$, mining honestly is optimal. ■

Remark 2 *Note that $\alpha > d_b$ holds in practical systems with block generation rate $\lambda_1 \ll \lambda_{\text{share}}$, where λ_{share} denotes the target share-submission rate with $\kappa_s \triangleq \frac{\lambda_1}{\lambda_{\text{share}}}$. Indeed, if*

$$d_b = \alpha, \tag{47}$$

then a miner with hashpower fraction α can submit shares only at the network block-generation rate (e.g., one share every 10 minutes in Bitcoin). Since pools typically require shares at least every few seconds, they choose a lower share difficulty,

$$d_b \sim \alpha\kappa_s. \tag{48}$$

Remark 3 *If the pool operator charges a fee $\varphi \in [0,1]$, i.e., a miner receives $(1-\varphi)R_s$ for each share worth R_s, then under a fixed difficulty ratio the post-DA adversarial revenue rate becomes $\alpha\frac{1-\varphi_e p}{1-\alpha p}$ with $\varphi_{\text{e}} = 1-(1-\varphi)(1-d_b)$, which is monotone in p. Hence, regardless of whether the honest baseline is solo (α) or pool mining ($(1-\varphi)\alpha$),*

the All-out Attack is optimal if and only if $\alpha > \varphi_{\mathrm{e}}$, and $p^* = 0$ otherwise. The resulting relative revenue change per epoch is $\frac{\alpha-\varphi_e}{1-\alpha}$ with respect to solo mining and $\frac{(\alpha-d_b)}{1-\alpha}$ with respect to pool mining.

Remark 4 *The analysis is presented for Bitcoin-style epoch-based DAAs for analytical simplicity. The attack is expected to be at least as effective under faster DA algorithms, since profitability arises immediately after the first downward adjustment in mining difficulty. Faster-reacting DAAs reduce the duration of the initial transient during which the attacker incurs negligible withholding losses, thereby increasing the attack's practical profitability.*

Remark 5 *Assuming $\alpha > d_b$ and neglecting the small fractional loss incurred by the adversary (i.e., assuming $d_b \approx 0$), the relative revenue change per epoch under AoA after the first DA for each mining entity is $\delta^{(1)} - 1$. Therefore, all mining entities benefit by the same magnitude per unit of hashpower they control. Conversely, the target pool operator incurs a loss of $\alpha \cdot \delta^{(1)}$ in each epoch. It is straightforward to show that the adversary can distribute its hashpower across multiple PPS mining pools without affecting the effectiveness of the attack (i.e., β may represent the combined power of multiple pools). For analytical simplicity, we consider a single target pool. Such a distribution may be preferred by the adversary to conceal the attack.*

Remark 6 *In longest-chain protocols without an uncle block mechanism, releasing withheld fPoW after honest miners mine a competing block induces a fork. This action marginally mitigates the pool operator's losses while simultaneously reducing the rewards received by external honest miners, since only one of the two blocks in the fork is ultimately rewarded. If the blockchain protocol incorporates uncle blocks into its DA mechanism, releasing withheld fPoW to induce a fork also reduces the magnitude of the difficulty decrease after the first epoch, in turn, diminishes adversarial profits.*

B. Variable Share Difficulty (Vardiff)

Theorem 5 *Under the PPS scheme with vardiff, the adversarial revenue change under BWH is maximized by*

$$p^* = \min\left\{1, \frac{1}{\alpha + 2\kappa_s}\right\}, \tag{49}$$

where $\kappa_s \triangleq \lambda_1/\lambda_{\mathrm{share}}$. In particular, if $\alpha < 1 - 2\kappa_s$, the optimal strategy is the All-out Attack.

The proof of Theorem 5, together with the derivation of the optimal revenue change slope, is provided in Appendix D.

Corollary 2 *Under the PPS scheme with vardiff, by using arbitrarily many sybil identities, the effective aggregate share submission rate can be made arbitrarily large, and hence the effective parameter κ_s arbitrarily small. Consequently, the All-out Attack is optimal, i.e., $p^* = 1$.*

Remark 7 *Under the standing assumption $\alpha + \beta < 0.5$, we have $\alpha < 0.5$. Hence, if $\kappa_s < \frac{1}{4}$, All-out Attack is the optimal strategy. Since practical blockchain systems satisfy $\lambda_1 \ll \lambda_{\mathrm{share}}$, we typically have $\kappa_s \ll 1$, and hence the All-out Attack is optimal in practice even without sybil identities. For example, in Bitcoin, if the pool targets one share per second,*

$$\kappa_s = \frac{1}{600} \approx 1.67 \times 10^{-3} \ll \frac{1}{4}, \tag{50}$$

which is more than two orders of magnitude smaller than the threshold. Even with a target share interval of one minute,

$$\kappa_s = \frac{60}{600} = 0.1 < \frac{1}{4}, \tag{51}$$

the All-out Attack remains optimal.

VII. Power Adjusting Hopping Attacks Under Block-Dependent Payout Schemes

In this section, we consider PAH attack that the adversary can use to increase its rewards. A primitive version of this attack for an adversary with negligible hashpower with optimal time to hop under PROP was derived in [2], [21]. Here, we extend the analysis of PAH to PROP, EXP and to any adversarial power size, where we allow partial power allocations. Under PAH, the adversary never withholds a fPoW, i.e., it submits all its pPoWs and fPoWs to the pool immediately, and only adjusts the fraction of its hashpower that it devotes to the target pool. Since every pool block is released immediately, every block is canonical and the round boundaries of the pool are simply the instants at which the pool finds a block. The adversary exploits the fact that, under round-based PROP and EXP, the value of a share depends on the cumulative blockwork of the open round.

Let t denote the time passed since the last pool block, which is public. Under PAH, the adversary mines in the pool with the fraction

$$p(t) = p_y \, \mathbf{1}\{t < t^*\} + p_o \, \mathbf{1}\{t \geq t^*\} \tag{52}$$

of α and solo with the rest, where $p_y, p_o \in [0,1]$ and $t^* \geq 0$ are the parameters of the attack; the subscripts y and o refer to young and old rounds. Similarly, let $a(t)$, $a'(t)$, $q(t)$ denote the quantities a_j, a'_j, q_j when the adversary mines with the power $p(t) = p_j$ with $j \in \{y, o\}$ at time t. Note that with $p_y = p_o = p$, the strategy reduces to the honest strategy that keeps the fraction p of its hashpower in the pool. Further, let

$$y^* = \lambda_1 a_y t^*, \qquad e^* = e^{-y^*}, \tag{53}$$

$$\chi^* = e^{-\lambda_d t^*}, \qquad \kappa_o = \frac{\lambda_1 a_o}{\lambda_d}. \tag{54}$$

We refer to the cumulative pool blockwork of the open round as its *blockwork age*, i.e., the blockwork age at time t is

$$y(t) = \int_0^t \lambda_1 a(t') \, dt', \tag{55}$$

and $y(t^*) = y^*$, i.e., the policy (52) is equivalently a threshold y^* on the blockwork age of the round. Let τ be the time when the pool block is found and the round ends, and $Y_\tau \triangleq y(\tau)$ the blockwork age of the round at τ. Since every hash is a valid block with the same probability, the chance that the pool finds a block depends only on how much blockwork it has done and not on how fast it was done. Hence, the power allocation of the adversary changes how fast the blockwork age of a round grows, but not the distribution of Y_τ, which is $\mathrm{Exp}(1)$ for every policy (see Lemma 8 in Appendix E).

Theorem 6 *Under round-based PROP and EXP, the block redundancy ratio of PAH is* $\delta^{\mathrm{PAH}} = 1$ *and*

$$\rho_A = \frac{n_S + r_h}{n_S + n_R + 1}, \tag{56}$$

$$\rho_P = \frac{\bar{r}_h}{n_S + n_R + 1}, \tag{57}$$

$$\rho_R = \frac{n_R}{n_S + n_R + 1}, \tag{58}$$

where $(r_h, \bar{r}_h)$*, are the reward fractions of the adversary and the honest target pool miners with* $\bar{r}_h = 1 - r_h$*, and*

$$n_R = \frac{\xi}{a_y}(1 - e^*) + \frac{\xi}{a_o} e^*, \tag{59}$$

$$n_S = \frac{\alpha q_y}{a_y}(1 - e^*) + \frac{\alpha q_o}{a_o} e^* \tag{60}$$

are the expected numbers of blocks mined by the honest miners outside the pool and the solo adversary per round, with

$$r_h^{\mathrm{PROP}} = a'_y(1 - e^*) + a'_o e^* + (a'_y - a'_o)\, y^* E_1(y^*), \tag{61}$$

$$r_h^{\mathrm{EXP}} = a'_y(1 - e^*) + e^* \kappa_o \int_0^1 v^{\kappa_o - 1} \frac{\alpha p_o + \big(\alpha p_y (1 - \chi^*) - \alpha p_o\big) v}{a_o + \big(a_y(1 - \chi^*) - a_o\big) v} \, dv. \tag{62}$$

The proof is provided in Appendix E. For $p_y = p_o = p$, both adversarial reward fractions reduce to r in (8), and Theorem 6 gives the revenue ratios of honest mining, $\rho_A = \alpha$, $\rho_P = \beta$ and $\rho_R = \xi$. As $\lambda_d \to 0$, EXP weighs all the shares of a round equally and $r_h^{\mathrm{EXP}} \to r_h^{\mathrm{PROP}}$. The goal of the adversary is to optimize an objective function, such as the revenue ratio ρ_A, with respect to p_y, p_o and t^*. Note that, under PAH, since $\delta^{\mathrm{PAH}} = 1$, there is no profit lag, i.e., if the parameters (p_y, p_o, t^*) yield $\rho_A > \alpha$, the gain of PAH is realized immediately and does not rely on a DA.

Remark 8 *Consider a pool share of blockwork $\epsilon \to 0$ submitted at blockwork age y under PROP by a negligible miner. In the pool, the share receives the fraction ϵ/Y_τ of the reward of the pool block, and given $Y_\tau > y$, the remaining blockwork age $Y_\tau - y$ is $\mathrm{Exp}(1)$ by Lemma 8(iii) and the memoryless property. Hence, the expected reward of the share is*

$$r(\epsilon) = \mathbb{E}\left[\frac{\epsilon}{Y_\tau} \,\middle|\, Y_\tau > y\right] = \int_0^\infty \frac{\epsilon\, e^{-u}}{y+u}\, du = \epsilon\, e^y E_1(y). \tag{63}$$

If mined solo, the same blockwork yields ϵ blocks in expectation, hence the value of the share relative to solo mining is $e^y E_1(y)$ which is exactly what Rosenfeld finds in [2, App. B]. It decreases from ∞ to 0, thus shares in young rounds are worth more than solo mining and shares in old rounds less, which is what PAH exploits. EXP reduces, but does not remove, this dependence of the share value on the blockwork age.

In Appendix E, we revisit the PAH-proofness of W-PPLNS [2]. Further, we consider a special case with $p_y = 1$ and $p_o = 0$ and optimize PAH only with respect to y^* for PROP and EXP. For PROP, by taking $\alpha \to 0$, we also recover the classical pool hopping result of [2], [21] for a negligible miner.

VIII. Numerical Results

In this section, we compare the vulnerability of the payout schemes to FAW, PAW, T-PAW, PAH and AoA. For every (α, β, γ), we maximize the adversarial revenue ratio ρ_A over (p_1, p_2, p_3, T) for T-PAW, over (p_1, p_2, p_3) for PAW, over p for FAW and over (p_y, p_o, t^*) for PAH, using the closed-form expressions of Sections V and VII with a multi-start interior-point method, and report the relative extra revenue at the optimum. Under PPS, the optimal attack is AoA (Section VI). Since the PPS operator pays the shares from its reserves, the revenue of the adversary is not a fraction of the block rewards, hence for AoA we report the relative revenue change per epoch after the first DA, i.e., $\frac{\alpha}{1-\alpha}$, which is directly comparable to the RER under the block-dependent schemes. We verify all analytical expressions with Monte Carlo simulations that model the attack cycles, the pool rounds, the PPLNS window and the EXP scores explicitly, without using any of the derivations. At the optimal parameters, the simulated revenue ratios, block redundancy ratios and reward fractions match the theory, with the reward fractions agreeing to the fourth digit for every scheme; see Appendix F. We set $\lambda_1 = 1$, which is without loss of generality since the results depend on T and t^* only through $\lambda_1 T$ and $\lambda_1 t^*$, and we assume a rational pool operator. Unless stated otherwise, we use $W = 2$ for PPLNS and $\lambda_d = 2\lambda_1$ for EXP, as in the original slush method. We evaluate all attacks on the grid $\alpha \in \{0.001, 0.01, 0.05\}$, $\beta \in \{0.05, 0.1\}$ and $\gamma \in \{0, 0.5\}$, and the quantitative statements in this section, such as "up to", "at most" and "at every point", refer to this grid.

A. FAW, PAW and T-PAW

Table IV compares the three withholding attacks. Since the adversary never changes its power allocation under FAW, all its reward fractions equal r in (8), and FAW yields the same gain under every scheme. The additional gain of the power adjustments, on the other hand, depends strongly on the scheme. Under PPLNS, the power adjustments improve on FAW only moderately, whereas under PROP, PAW already improves on FAW considerably. Under EXP, which weighs the recent shares more, the deadline of T-PAW is decisive: for $(\alpha, \beta, \gamma) = (0.05, 0.05, 0)$, T-PAW yields 17.5 times the gain of PAW. Under PROP, the deadline of T-PAW improves the long-run gain by up to 1.46 times compared to PAW, and its effect is even stronger on the short-term gain: the largest pre-adjustment gain of T-PAW is up to 4.5 times that of PAW (Section VIII-B). In all schemes, the gains increase with γ, as the released fPoWs win more fork races.

Prior work quantified the improvement of PAW over FAW only under GRP, where PAW was shown to earn up to 2.5 times the extra revenue of FAW [10]. Table IV shows that the improvements of PAW over FAW and of T-PAW

α	β	FAW	PPLNS		PROP		EXP		GRP	
			PAW	T-PAW	PAW	T-PAW	PAW	T-PAW	PAW	T-PAW
$\gamma = 0$										
0.001	0.05	0.0013	0.0013	0.0015	0.0036	0.0046	0.0035	0.1080	0.0013	0.0737
0.001	0.1	0.0027	0.0029	0.0032	0.0061	0.0061	0.0067	0.0612	0.0027	0.0402
0.01	0.05	0.0133	0.0139	0.0157	0.0351	0.0511	0.0326	0.8895	0.0135	0.6159
0.01	0.1	0.0279	0.0303	0.0333	0.0606	0.0711	0.0642	0.5575	0.0279	0.3671
0.05	0.05	0.0706	0.0739	0.0834	0.1603	0.1942	0.1363	2.3820	0.0781	1.6976
0.05	0.1	0.1476	0.1602	0.1766	0.2859	0.2977	0.2739	1.9553	0.1516	1.3137
$\gamma = 0.5$										
0.001	0.05	0.0025	0.0038	0.0038	0.0179	0.0179	0.0867	0.1694	0.0506	0.1107
0.001	0.1	0.0050	0.0073	0.0073	0.0201	0.0201	0.0553	0.0924	0.0311	0.0587
0.01	0.05	0.0252	0.0381	0.0385	0.1590	0.1633	0.6896	1.3657	0.4129	0.9064
0.01	0.1	0.0499	0.0736	0.0743	0.1869	0.1898	0.4954	0.8355	0.2826	0.5342
0.05	0.05	0.1303	0.1928	0.1952	0.5295	0.5442	1.6722	3.4247	1.0852	2.3847
0.05	0.1	0.2574	0.3721	0.3759	0.7117	0.7240	1.6537	2.8003	0.9978	1.8491

TABLE IV: Optimal RER $\rho_A/\alpha - 1$ (%) under FAW, PAW and T-PAW for every block-dependent payout scheme, with $W = 2$ for PPLNS and $\kappa_1 = \frac{1}{2}$ for EXP.

over PAW depend strongly on the scheme. PAW yields at most 1.52 times the gain of FAW under PPLNS, whereas it yields up to 7.2 and 34.7 times the gain of FAW under PROP and EXP, respectively, for $\gamma = 0.5$. Without fork races, i.e., $\gamma = 0$, PAW barely improves on FAW under PPLNS and GRP, at most 1.09 and 1.11 times, respectively. On the other hand, T-PAW improves on PAW by up to more than an order of magnitude under EXP and GRP for $\gamma = 0$, and about twofold for $\gamma = 0.5$, but at most 1.46 times under PPLNS and PROP.

Remark 9 *GRP, which is adopted by prior analyses [10]–[12], is not a reliable proxy for the schemes used in practice, and the direction of its error depends on the scheme. Under PROP, the blockwork carried over from earlier attack cycles dilutes the power adjustments, and GRP overstates the T-PAW gain by up to more than an order of magnitude, e.g.,* 0.0737% *versus* 0.0046% *for* $(\alpha, \beta, \gamma) = (0.001, 0.05, 0)$. *Under EXP, the decay emphasizes the recent shares of the withholding phase, and GRP understates the T-PAW gain, e.g.,* 1.6976% *versus* 2.3820% *for* $(\alpha, \beta, \gamma) = (0.05, 0.05, 0)$ *in Table IV.*

B. Profit Lag

The relative extra revenue measures the gain after the DA. Before it, an attack that wastes hashpower, i.e., $\delta > 1$, earns ρ_A per epoch of duration $\delta\tau_0$, hence it profits before the adjustment only if $\rho_A > \alpha\delta$, in which case its profit lag is zero. Otherwise, its profit lag is $\mathcal{T}_A^{S,H} = \frac{\rho_A(\delta-1)}{\rho_A-\alpha}\tau_0$, which exceeds the duration $\delta\tau_0$ of the first epoch.

Table V reports the relative revenue change at the end of the first epoch $\overline{\Delta}_A^{S,H}(t_1) = \frac{\rho_A}{\alpha} - \delta$ and the profit lag of T-PAW at the parameters that maximize ρ_A. Under EXP (and GRP), T-PAW profits before the DA at every point, and the pre-adjustment gain is a large part of the long-run gain, e.g., 2.0712% versus 2.3820% under EXP for $(\alpha, \beta, \gamma) = (0.05, 0.05, 0)$. Under PROP, the pre-adjustment gain is small and may be negative, e.g., for $(\alpha, \beta, \gamma) = (0.001, 0.1, 0)$ and $(0.05, 0.1, 0.5)$, where the profit lag is $1.32\tau_0$ and $1.04\tau_0$, respectively. Under PPLNS, the pre-adjustment gain is negative at every point, and the profit lag is between $1.59\tau_0$ and $1.73\tau_0$.

An adversary that targets a short-term gain may instead maximize its revenue change at the end of the first epoch, $\Delta_A^{S,H}(t_1) = \rho_A - \alpha\delta$. Table VI shows that, under PROP and PPLNS, the resulting strategy rarely withholds, i.e., it has a small p_1 and $\delta \approx 1$, but it moves almost all of its power to the pool when it does, with a short deadline. Under PROP, such a strategy profits before the DA at every point, but its pre-adjustment gain stays below 0.25%, and it sacrifices part of the long-run gain, e.g., 0.5160% versus 0.7240% for $(\alpha, \beta, \gamma) = (0.05, 0.1, 0.5)$. PAW, i.e., $T \to \infty$, also profits before the DA under PROP at every point, but with a pre-adjustment gain up

α	β	PPLNS RER	PPLNS $\overline{\Delta}_A(t_1)$	PPLNS $\mathcal{T}_A$	PROP RER	PROP $\overline{\Delta}_A(t_1)$	PROP $\mathcal{T}_A$	EXP RER	EXP $\overline{\Delta}_A(t_1)$	EXP $\mathcal{T}_A$	GRP RER	GRP $\overline{\Delta}_A(t_1)$	GRP $\mathcal{T}_A$
$\gamma = 0$													
0.001	0.05	0.0015	-0.0009	1.63	0.0046	0.0006	0	0.1080	0.0985	0	0.0737	0.0673	0
0.001	0.1	0.0032	-0.0022	1.68	0.0061	-0.0019	1.32	0.0612	0.0506	0	0.0402	0.0330	0
0.01	0.05	0.0157	-0.0111	1.71	0.0511	0.0162	0	0.8895	0.8039	0	0.6159	0.5576	0
0.01	0.1	0.0333	-0.0234	1.70	0.0711	0.0116	0	0.5575	0.4562	0	0.3671	0.2984	0
0.05	0.05	0.0834	-0.0605	1.73	0.1942	0.0366	0	2.3820	2.0712	0	1.6976	1.4728	0
0.05	0.1	0.1766	-0.1285	1.73	0.2977	0.0145	0	1.9553	1.5218	0	1.3137	1.0077	0
$\gamma = 0.5$													
0.001	0.05	0.0038	-0.0023	1.60	0.0179	0.0057	0	0.1694	0.1531	0	0.1107	0.0987	0
0.001	0.1	0.0073	-0.0044	1.60	0.0201	0.0022	0	0.0924	0.0747	0	0.0587	0.0456	0
0.01	0.05	0.0385	-0.0226	1.59	0.1633	0.0543	0	1.3657	1.2183	0	0.9064	0.7968	0
0.01	0.1	0.0743	-0.0439	1.59	0.1898	0.0259	0	0.8355	0.6666	0	0.5342	0.4076	0
0.05	0.05	0.1952	-0.1240	1.64	0.5442	0.0800	0	3.4247	2.8882	0	2.3847	1.9678	0
0.05	0.1	0.3759	-0.2419	1.65	0.7240	-0.0259	1.04	2.8003	2.0795	0	1.8491	1.2925	0

TABLE V: T-PAW at the parameters that maximize ρ_A under every block-dependent payout scheme: RER (%), relative revenue change at the end of the first epoch $\overline{\Delta}_A(t_1)$ (%) and profit lag $\mathcal{T}_A$ in units of τ_0.

		$\gamma = 0$				$\gamma = 0.5$			
		PPLNS		PROP		PPLNS		PROP	
α	β	RER	$\overline{\Delta}_A(t_1)$	RER	$\overline{\Delta}_A(t_1)$	RER	$\overline{\Delta}_A(t_1)$	RER	$\overline{\Delta}_A(t_1)$
0.001	0.05	–	–	0.0037	0.0013	0.0013	<0.0001	0.0161	0.0091
0.001	0.1	0.0003	<0.0001	0.0036	0.0008	0.0031	0.0002	0.0156	0.0075
0.01	0.05	0.0054	0.0008	0.0437	0.0246	0.0152	0.0024	0.1408	0.0786
0.01	0.1	0.0114	0.0017	0.0566	0.0295	0.0290	0.0045	0.1482	0.0742
0.05	0.05	0.0282	0.0043	0.1566	0.0828	0.0737	0.0115	0.4286	0.2181
0.05	0.1	0.0596	0.0090	0.2240	0.1114	0.1403	0.0218	0.5160	0.2433

TABLE VI: T-PAW optimized for the revenue change at the end of the first epoch, $\Delta_A(t_1) = \rho_A - \alpha\delta$, under PPLNS ($W = 2$) and PROP: RER (%) and relative revenue change $\overline{\Delta}_A(t_1)$ (%). The profit lag is zero at every point, except for the point marked by – where no T-PAW strategy profits before the DA.

to 4.5 times smaller than that of T-PAW, e.g., 0.0199% versus 0.0828% for $(\alpha, \beta, \gamma) = (0.05, 0.05, 0)$. Under PPLNS, the pre-adjustment gain of T-PAW is at most 0.0218%, and no T-PAW strategy profits before the DA for $(\alpha, \beta, \gamma) = (0.001, 0.05, 0)$. The deadline T matters even more under PPLNS, where, for $\gamma = 0$, the largest pre-adjustment gain of T-PAW is more than an order of magnitude larger than that of PAW. In contrast, PAH does not withhold any block, thus $\delta^{\mathrm{PAH}} = 1$ and its profit lag is zero whenever $\rho_A > \alpha$.

C. Which Attack Dominates Under Each Scheme

Table VII compares the strongest attacks against each scheme. Under PROP, PAH dominates by far: its gain is between 17.6027% and 27.9685%, whereas the gain of T-PAW stays below 1%. For a small adversary, the PAH gain approaches the 28% of a negligible miner [2], [21], e.g., 27.7903% for $(\alpha, \beta) = (0.001, 0.05)$. As α/β grows, the adversary dilutes the young rounds with its own shares and the gain decreases, e.g., to 17.6027% for $\alpha = \beta = 0.05$. Under EXP, both attacks are effective and the best attack depends on the parameters: PAH is preferable unless the adversary is large relative to the pool, e.g., 3.8373% versus at most 0.8355% for $(\alpha, \beta) = (0.01, 0.1)$, whereas T-PAW is preferable for $\alpha = \beta = 0.05$, e.g., 3.4247% versus 1.5229% for $\gamma = 0.5$, and 2.3820% versus 1.5229% for $\gamma = 0$ (Table IV). Under PPLNS, PAH yields no gain by Proposition 1, and the gain of T-PAW is the smallest

α	β	PPS AoA	PAH PROP	PAH EXP	T-PAW, $\gamma = 0.5$ PPLNS	T-PAW, $\gamma = 0.5$ PROP	T-PAW, $\gamma = 0.5$ EXP
0.001	0.05	0.1001	27.7903	2.1842	0.0038	0.0179	0.1694
0.001	0.1	0.1001	27.9685	4.0099	0.0073	0.0201	0.0924
0.01	0.05	1.0101	24.9790	2.0093	0.0385	0.1633	1.3657
0.01	0.1	1.0101	26.4544	3.8373	0.0743	0.1898	0.8355
0.05	0.05	5.2632	17.6027	1.5229	0.1952	0.5442	3.4247
0.05	0.1	5.2632	21.4961	3.2522	0.3759	0.7240	2.8003

TABLE VII: Optimal RER $\rho_A/\alpha - 1$ (%) of the strongest attacks against each payout scheme: AoA under PPS, i.e., $\frac{\alpha}{1-\alpha}$, which does not depend on β or γ, PAH under PROP and EXP, which does not depend on γ, and T-PAW with $\gamma = 0.5$, with $W = 2$ for PPLNS and $\kappa_1 = \frac{1}{2}$ for EXP.

α	β	T-PAW, $\gamma = 0$, $\lambda_d =$ λ_1	$2\lambda_1$	$4\lambda_1$	T-PAW, $\gamma = 0.5$, $\lambda_d =$ λ_1	$2\lambda_1$	$4\lambda_1$	PAH λ_1	$2\lambda_1$	$4\lambda_1$
0.001	0.05	0.056	0.108	0.180	0.102	0.169	0.247	3.990	2.184	1.153
0.001	0.1	0.035	0.061	0.098	0.060	0.092	0.130	6.921	4.010	2.195
0.01	0.05	0.482	0.890	1.447	0.853	1.366	1.959	3.667	2.009	1.061
0.01	0.1	0.324	0.558	0.876	0.554	0.836	1.165	6.617	3.837	2.102
0.05	0.05	1.438	2.382	3.579	2.312	3.425	4.666	2.768	1.523	0.805
0.05	0.1	1.213	1.955	2.918	1.946	2.800	3.774	5.585	3.252	1.785

TABLE VIII: Optimal relative extra revenue $\rho_A/\alpha - 1$ (%) under T-PAW and PAH for EXP with different decay rates λ_d.

among all schemes, at most 0.3759%. However, it remains positive for every (α, β, γ), thus PPLNS mitigates T-PAW but does not eliminate it.

Comparison with PPS: Under PPS, each share is paid immediately and its payment does not depend on the rounds, therefore PAH yields no gain, whereas BWH is profitable after the DA, where the optimal strategy is AoA with a gain of $\frac{\alpha}{1-\alpha}$ per epoch that is borne by the pool operator. Since the pre-adjustment revenue change of AoA is slightly negative by Lemma 1, its profit lag is slightly above the duration $\frac{\tau_0}{1-\alpha}$ of the first epoch. For $\alpha = 0.05$, the AoA gain under PPS exceeds the gain of every attack under every block-dependent scheme with $W = 2$ and $\lambda_d = 2\lambda_1$, except PAH under PROP. For a small adversary, however, block-dependent schemes are not safer: T-PAW under EXP already exceeds the AoA gain, e.g., 1.3657% versus 1.0101% for $(\alpha, \beta, \gamma) = (0.01, 0.05, 0.5)$, and PAH under PROP and EXP exceeds it by far, since the relative gain of PAH barely decreases as $\alpha \to 0$, while the AoA gain vanishes linearly in α. Among all schemes we consider, PPLNS is the least vulnerable for every (α, β, γ).

D. Decay Rate of EXP

Table VIII reveals a trade-off in the design of EXP. As λ_d increases, the scores of the shares submitted before the withholding phase decay faster, hence the power adjustments of T-PAW weigh more, and the T-PAW gain increases, e.g., from 1.438% to 3.579% for $(\alpha, \beta, \gamma) = (0.05, 0.05, 0)$ as λ_d increases from λ_1 to $4\lambda_1$. At the same time, the score of an old round saturates faster, thus the value of a share depends less on the blockwork age of the round, and the PAH gain decreases, e.g., from 2.768% to 0.805% for the same parameters. In the limit $\lambda_d \to 0$, EXP reduces to PROP, which is the most vulnerable scheme to PAH. EXP cannot protect against both attacks by tuning λ_d: a slow decay invites PAH and a fast one invites T-PAW.

E. Window of PPLNS

Table IX shows that the T-PAW gain under PPLNS decreases with the window size W, e.g., from 0.5232% to 0.3137% for $(\alpha, \beta, \gamma) = (0.05, 0.1, 0.5)$ as W increases from 1 to 4. By Theorem 1, the power adjustments of

		$\gamma = 0$, $W =$			$\gamma = 0.5$, $W =$		
α	β	1	2	4	1	2	4
0.001	0.05	0.0019	0.0015	0.0013	0.0054	0.0038	0.0029
0.001	0.1	0.0041	0.0032	0.0028	0.0101	0.0073	0.0060
0.01	0.05	0.0215	0.0157	0.0139	0.0560	0.0385	0.0314
0.01	0.1	0.0452	0.0333	0.0294	0.1049	0.0743	0.0614
0.05	0.05	0.1142	0.0834	0.0738	0.2796	0.1952	0.1606
0.05	0.1	0.2383	0.1766	0.1565	0.5232	0.3759	0.3137

TABLE IX: Optimal relative extra revenue $\rho_A/\alpha - 1$ (%) under T-PAW for W-PPLNS with different window sizes W.

T-PAW affect the reward fractions only through terms of order $1/W$, since the blockwork of a single attack cycle is a small part of the window. The part of the gain that stems from the fork races, as in FAW, does not depend on W, therefore the gain does not vanish as W grows but approaches that of FAW, e.g., 0.0738% for $W = 4$ versus 0.0706% under FAW for $(\alpha, \beta, \gamma) = (0.05, 0.05, 0)$ (Table IV). Hence, a longer window alone cannot make PPLNS fully resistant to T-PAW.

IX. Conclusion

We analyzed block withholding and pool hopping attacks under both share-based and block-dependent pool payout schemes, taking the DA into account. We introduced T-PAW, which contains FAW and PAW as special cases and improves on them, and derived the revenue ratios under W-PPLNS, round-based PROP and EXP, and we showed that GRP, which is adopted by prior analyses, is not a reliable proxy, since it overstates the gain under PROP and understates it under EXP. The deadline of T-PAW improves on PAW by up to more than an order of magnitude under EXP and GRP when fork races are unfavorable, but only moderately under PROP and PPLNS. Moreover, T-PAW profits before the DA under EXP and GRP, whereas under PROP and PPLNS its pre-adjustment gain may exist but to a much smaller degree. We further introduced PAH, which requires neither withholding nor a DA, and derived its revenue ratios for an adversary of any size. Since PAH does not waste any hashpower, its gain is realized immediately, i.e., its profit lag is zero.

We further showed that PPS pools are not incentive compatible under BWH once the DA is taken into account. The optimal attack is AoA, which raises the adversarial revenue per epoch from α to $\frac{\alpha}{1-\alpha}$ after the first adjustment at a negligible loss before it, while the target pool operator bears all losses. Remarkably, although the mechanisms differ, this gain coincides with the upper bound of optimal selfish mining with adversarial network influence $\gamma = 1$ [41], where every adversarial block effectively invalidates a block mined by another entity.

Comparing the schemes, each one is vulnerable to a different attack. PROP is highly vulnerable to PAH, and EXP cannot be tuned against both attacks, as a slow decay invites PAH and a fast decay invites T-PAW. PPS and PPLNS are immune to PAH, but PPS is vulnerable to AoA, whose gain grows with α and, for a large adversary, exceeds that of every attack under every block-dependent scheme except PAH under PROP, whereas for a small adversary EXP and PROP are more exposed. PPLNS yields the smallest withholding gain among all schemes, including AoA under PPS, and is therefore the most robust scheme we consider.

On the defense side, under AoA the target pool discovers blocks at only a fraction $\frac{\beta}{\alpha+\beta}$ of its expected rate, a deviation that the pool operator may statistically detect over an epoch; distributing the attack across multiple pools dilutes this deviation per pool, while multiple identities obscure its source. Since the largest Bitcoin pools operate under PPS/FPPS, these results call for pool-side detection mechanisms and payout designs that account for the DA. Further directions include strategies that combine withholding and hopping, which might prove especially useful against score-based schemes, and attacks on multiple target pools, which we expect to increase the gain especially for PAH.

Appendix A
Reward Fractions for W-PPLNS

We first consider the blockwork of the pool accumulated in an attack cycle. Note that blockwork refers to the expected number of fPoWs generated:

1) *First phase:* Let $\hat{Q}_1$ denote the blockwork of a pool in the first phase of an attack cycle which lasts $T_1 \sim \text{Exp}(\lambda_1)$, where the pool works with blockwork rate $\lambda_1 a_1$. Hence, we have $\hat{Q}_1 \triangleq \lambda_1 a_1 T_1 \sim \text{Exp}(1/a_1)$, with $\mathbb{E}[\hat{Q}_1] = a_1$ and $\mathbb{E}[\hat{Q}_1^2] = 2a_1^2$.
2) *Withholding phase:*
   a) *Successful (Case 4a)*: With probability θ the withholding deadline is reached and we let $\hat{T}$ denote the blockwork of the pool. This withholding phase lasts T, where the pool works with blockwork rate $\lambda_1 a_2$. Hence, we have $\hat{T} \triangleq \lambda_1 a_2 T$ which is deterministic.
   b) *Unsuccessful (Case 4b)*: With probability $1-\theta$ the phase ends earlier after the time T_2', and the blockwork of the pool is $\hat{Q}_2 \triangleq \lambda_1 a_2 T_2'$. Since $T_2' \sim (T_2 \mid T_2 < T)$ with $T_2 \sim \text{Exp}(\lambda_2)$, we have $\hat{Q}_2 \sim (J_2 \mid J_2 < \hat{T})$ with $J_2 \sim \text{Exp}(\mu_2)$.
3) *Fork race phase:* When the withholding phase is ended before the deadline by a block of the honest miners outside the pool, a fork race of duration $T_3 \sim \text{Exp}(\lambda_1)$ follows with the pool at hash fraction a_3, hence the blockwork of the pool is $\hat{Q}_3 \triangleq \lambda_1 a_3 T_3 \sim \text{Exp}(1/a_3)$, with $\mathbb{E}[\hat{Q}_3] = a_3$ and $\mathbb{E}[\hat{Q}_3^2] = 2a_3^2$.

It is straightforward to verify that

$$\nu_1 \triangleq \mathbb{E}[\hat{Q}_2] = \frac{1}{\mu_2} - \frac{\theta \hat{T}}{1-\theta}, \tag{64}$$

$$\nu_2 \triangleq \mathbb{E}[\hat{Q}_2^2] = \frac{2}{\mu_2^2} - \frac{\theta \hat{T}}{1-\theta}\left(\hat{T} + \frac{2}{\mu_2}\right), \tag{65}$$

Events	Probability	Pool blockwork S	Adversarial pool blockwork X
Cases 1,2,3	$1-\alpha p_1$	$\hat{Q}_1$	$a_1'\hat{Q}_1$
Case 4a	$\alpha p_1 \theta$	$\hat{Q}_1 + \hat{T}$	$a_1'\hat{Q}_1 + a_2'\hat{T}$
Case 4b-i,4b-ii	$\alpha p_1(1-\theta)(\pi_A+\pi_P)$	$\hat{Q}_1 + \hat{Q}_2$	$a_1'\hat{Q}_1 + a_2'\hat{Q}_2$
Case 4b-iii	$\alpha p_1(1-\theta)\pi_R$	$\hat{Q}_1 + \hat{Q}_2 + \hat{Q}_3$	$a_1'\hat{Q}_1 + a_2'\hat{Q}_2 + a_3'\hat{Q}_3$

TABLE X: Cumulative pool and adversarial pool blockworks S, X in an attack cycle.

Let $S = \hat{Q}_1 + V$ denote the cumulative blockwork of the pool at the end of an attack cycle. Table X lists the events of an attack cycle, their probabilities, the pool blockwork S of the cycle and its adversarial part X. We have

$$V = \begin{cases} 0, & \text{w.p. } 1-\alpha p_1, \\ \hat{T}, & \text{w.p. } \alpha p_1 \theta, \\ \hat{Q}_2, & \text{w.p. } \alpha p_1(1-\theta)(\pi_A+\pi_P), \\ \hat{Q}_2 + \hat{Q}_3, & \text{w.p. } \alpha p_1(1-\theta)\pi_R. \end{cases} \tag{66}$$

Using $\pi_A + \pi_P + \pi_R = 1$ and $\mathbb{E}[(\hat{Q}_2 + \hat{Q}_3)^2] = \nu_2 + 2a_3\nu_1 + 2a_3^2$,

$$\mathbb{E}[V] = \alpha p_1 \left\{ \theta \hat{T} + (1-\theta)\left(\nu_1 + \pi_R a_3\right) \right\}, \tag{67}$$

$$\mathbb{E}[V^2] = \alpha p_1 \left\{ \theta \hat{T}^2 + (1-\theta)\left(\nu_2 + 2\pi_R a_3 \nu_1 + 2\pi_R a_3^2\right) \right\}. \tag{68}$$

The cumulative blockwork of the pool at the end of an attack cycle is $S = \hat{Q}_1 + V$ with $\hat{Q}_1$ independent of V, thus

$$\mathbb{E}[S] = a_1 + \mathbb{E}[V],\ \mathbb{E}[S^2] = 2a_1^2 + 2a_1\mathbb{E}[V] + \mathbb{E}[V^2]. \tag{69}$$

Here, explicitly

$$\mathbb{E}[S] = a_1 + \alpha p_1 (1-\theta)\frac{a_2 + a_3\xi}{1-\alpha p_2}. \tag{70}$$

Let $b(t) = \sigma$ denote the cumulative blockwork of the target pool since the beginning until time t. Under W-PPLNS the reward of a pool block that the operator receives at t is split according to the blockworks submitted

during $(b^{-1}(\sigma-W), b^{-1}(\sigma)]$. Let $x(\sigma)$ be the adversarial fraction of the blockwork submitted within an infinitesimal neighborhood of $b^{-1}(\sigma)$. In a T-PAW cycle, we have

$$x(\sigma) = \begin{cases} a'_1, & \text{first phase,} \\ a'_2, & \text{withholding phase,} \\ a'_3, & \text{fork race phase.} \end{cases} \tag{71}$$

Hence, among the blockwork submitted during $(b^{-1}(\sigma - W), b^{-1}(\sigma)]$, the adversarial blockwork is

$$\frac{1}{W}\int_{\sigma-W}^{\sigma} x(s)\, ds. \tag{72}$$

Let $X = \int_0^S x(\sigma)d\sigma$ be the adversarial pool blockwork in an attack cycle. It is straightforward to verify that

$$\begin{aligned} \mathbb{E}[X] &= \alpha p_1 + \alpha p_1 a'_2 \left\{\theta\hat{T} + (1-\theta)\nu_1\right\} + \alpha p_1(1-\theta)\pi_R \alpha p_3 \\ &= \alpha p_1\Big(1 + (1-\theta)\frac{\alpha p_2 + \alpha p_3 \xi}{1-\alpha p_2}\Big). \end{aligned} \tag{73}$$

Thus, the long-run adversarial blockwork fraction is the ratio of (73) and (70),

$$\bar{x} \triangleq \frac{\mathbb{E}[X]}{\mathbb{E}[S]} = \frac{\alpha p_1\left[1 - \alpha p_2 + (1-\theta)\big(\alpha p_2 + \alpha p_3 \xi\big)\right]}{a_1(1-\alpha p_2) + \alpha p_1(1-\theta)\left[a_2 + a_3\xi\right]}. \tag{74}$$

Next, we find $\mathbb{E}[\Psi] \triangleq \mathbb{E}[\int_0^S x(\sigma)(S-\sigma)\, d\sigma]$ by splitting the integral over the three phases.

- First phase, $\sigma \in [0, \hat{Q}_1]$, $x = a'_1$, $S - \sigma = \hat{Q}_1 - \sigma + V$, $a'_1 \int_0^{\hat{Q}_1}(\hat{Q}_1 - \sigma + V)d\sigma = a'_1\left(\frac{\hat{Q}_1^2}{2} + \hat{Q}_1 V\right)$, with expectation $a'_1\left(a_1^2 + a_1\mathbb{E}[V]\right) = \alpha p_1\left(a_1 + \mathbb{E}[V]\right)$ since $\hat{Q}_1$ and V are independent.
- Withholding phase of pool work L followed by pool work R until the end of the cycle, $x = a'_2$: $a'_2\int_0^L (L - s + R)ds = a'_2\left(\frac{L^2}{2} + LR\right)$. In Case 4a, $L = \hat{T}$ and $R = 0$. In Case 4b, $L = \hat{Q}_2$, and $R = \hat{Q}_3$ with probability π_R and $R = 0$ otherwise, thus $\mathbb{E}[L^2/2 + LR] = \nu_2/2 + \pi_R a_3 \nu_1$.
- Fork race phase, $x = a'_3$, and the cycle ends with the race: $a'_3\int_0^{\hat{Q}_3}(\hat{Q}_3 - s)ds = a'_3\hat{Q}_3^2/2$, with expectation $a'_3 a_3^2 = \alpha p_3\, a_3$.

With the probabilities of each phase happening,

$$\mathbb{E}[\Psi] = \alpha p_1\left(a_1 + \mathbb{E}[V]\right) + \alpha p_1(1-\theta)\pi_R\, \alpha p_3\, a_3 + \alpha p_1 a'_2\left\{\frac{\theta\hat{T}^2}{2} + (1-\theta)\left(\frac{\nu_2}{2} + \pi_R a_3 \nu_1\right)\right\}. \tag{75}$$

A. Adversarial Blockwork in the Window Before a Cycle

Let $H(v)$ be the expected adversarial pool blockwork in a blockwork window v that immediately precedes the start of an attack cycle. Throughout this section $m_1 \triangleq \mathbb{E}[S]$, $m_2 \triangleq \mathbb{E}[S^2]$ and F is the distribution function of S.

Lemma 2 *H satisfies the renewal equation*

$$H(v) = \mathbb{E}\left[\int_{(S-v)^+}^{S} x(\sigma)\, d\sigma\right] + \mathbb{E}\big[H(v-S)\,\mathbf{1}\{S < v\}\big], \tag{76}$$

and, as $v \to \infty$,

$$H(v) = \bar{x}\, v + H_0 + o(1), \qquad H_0 \triangleq \frac{\bar{x}\,\mathbb{E}[S^2]}{2\,\mathbb{E}[S]} - \frac{\mathbb{E}[\Psi]}{\mathbb{E}[S]}. \tag{77}$$

Proof: *Step 1, the renewal equation:* Attack cycles are i.i.d., hence the cycles that precede a given cycle start, read backwards, are i.i.d. copies of $(S, x(\cdot))$ and are independent of the cycle that starts there. Consider the last cycle before the start and let S be its pool blockwork. If $S \geq v$, the pool blockwork window v lies inside this cycle and covers its positions $(S - v, S]$. If $S < v$, the pool blockwork window contains the whole cycle, and the remaining

pool blockwork window $v-S$ ends at the start of that cycle, which is again a cycle start, thus it holds $H(v-S)$ pool blockwork in expectation. Both cases are written in (76). With $h_0(v) \triangleq \mathbb{E}\big[\int_{(S-v)^+}^{S} x(\sigma)d\sigma\big]$, (76) gives

$$H(v) = h_0(v) + \int_0^v H(v-s)\,dF(s), \tag{78}$$

and

$$H(v) = \int_{[0,v]} h_0(v-y)\,dU(y),\ U(y) \triangleq \sum_{n=0}^{\infty} F^{*n}(y). \tag{79}$$

U is the renewal function and $U(y)$ is the expected number of cycle starts in a pool blockwork window v before the given cycle start, counting the given one ($F^{*0} = \mathbf{1}\{y \geq 0\}$). Next, let $\hat{h}(v)$ be the expected adversarial pool blockwork of an attack cycle that lies more than v before its end,

$$h_0(v) = \mathbb{E}[X] - \hat{h}(v),\ \hat{h}(v) \triangleq \mathbb{E}\bigg[\int_0^{(S-v)^+} x(\sigma)\,d\sigma\bigg]. \tag{80}$$

Inserting in (79),

$$H(v) = \underbrace{\mathbb{E}[X]\,U(v)}_{\text{(I)}} - \underbrace{\int_{[0,v]} \hat{h}(v-y)\,dU(y)}_{\text{(II)}}. \tag{81}$$

Term (I) counts all adversarial pool blockwork of every cycle that has its later boundary inside the interval. The earliest of these cycles extends beyond the interval, and term (II) removes its shares that lie outside. We will later use the following fact (see, [42]): If F is non-lattice with mean m_1 and $z \geq 0$ is non-increasing with $\int_0^\infty z(t)dt < \infty$, then

$$\int_{[0,v]} z(v-y)\,dU(y) \ \xrightarrow{v\to\infty}\ \frac{1}{m_1}\int_0^\infty z(t)\,dt. \tag{82}$$

Step 2, term (I): Let $g(v) \triangleq U(v) - v/m_1$. Note $U = 1 + U * F$ and substitute $U(v) = g(v) + v/m_1$ on both sides,

$$g(v) = \underbrace{1 - \frac{v}{m_1} + \frac{1}{m_1}\int_0^v (v-s)\,dF(s)}_{\triangleq\, z(v)} + \int_0^v g(v-s)\,dF(s). \tag{83}$$

By exchanging the order of integration, $\int_0^v (v-s)\,dF(s) = \int_0^v \int_s^v dt\,dF(s) = \int_0^v F(t)\,dt$, hence

$$z(v) = 1 - \frac{1}{m_1}\int_0^v \big(1-F(t)\big)dt = \frac{1}{m_1}\int_v^\infty \big(1-F(t)\big)dt. \tag{84}$$

Thus, $z \geq 0$ is non-increasing, and

$$\int_0^\infty z(v)\,dv = \frac{1}{m_1}\int_0^\infty \int_v^\infty \big(1-F(t)\big)dt\,dv \tag{85}$$

$$= \frac{1}{m_1}\int_0^\infty t\big(1-F(t)\big)dt = \frac{m_2}{2m_1}. \tag{86}$$

The renewal equation $g = z + g * F$ has the solution $g = z * U$, and (82) gives $g(v) \to m_2/(2m_1^2)$, i.e.,

$$U(v) = \frac{v}{m_1} + \frac{m_2}{2m_1^2} + o(1). \tag{87}$$

Step 3, term (II): The function $\hat{h}$ in (80) is non-negative and non-increasing in v. A pool blockwork at position σ of a cycle is counted in $\hat{h}(v)$ exactly when $\sigma < S - v$, i.e., for $v \in [0, S-\sigma)$, hence by exchanging the order of

integration

$$\int_0^\infty \hat{h}(v)\,dv = \mathbb{E}\Big[\int_0^S x(\sigma)\int_0^\infty \mathbf{1}\{v < S-\sigma\}\,dv\,d\sigma\Big] \tag{88}$$

$$= \mathbb{E}\Big[\int_0^S x(\sigma)(S-\sigma)\,d\sigma\Big] = \mathbb{E}[\Psi] < \infty. \tag{89}$$

Hence, (82) applies with $z = \hat{h}$ and term (II) converges to $\mathbb{E}[\Psi]/m_1$. Combining (I) and (II) completes the proof. ■

B. Reward Fractions of the Adversary

Theorem 7 *If a canonical pool block received when the cumulative blockwork since time immemorial is $\sigma > W$, with*

$$\mathcal{A}(\hat{q}_1,\hat{q}_2,\hat{q}_3) \triangleq a_3'\min\{\hat{q}_3, W\} + a_2'\min\{\hat{q}_2, (W-\hat{q}_3)^+\} + a_1'\min\{\hat{q}_1, (W-\hat{q}_2-\hat{q}_3)^+\} + H\big((W-\hat{q}_1-\hat{q}_2-\hat{q}_3)^+\big), \tag{90}$$

the expected adversarial shares under W-PPLNS are

$$r_1 = \frac{\mathbb{E}[\mathcal{A}(\hat{Q}_1,0,0)]}{W}, \tag{91}$$

$$r_s = \frac{\mathbb{E}[\mathcal{A}(\hat{Q}_1,\hat{T},0)]}{W}, \tag{92}$$

$$r_u = \frac{\mathbb{E}[\mathcal{A}(\hat{Q}_1,\hat{Q}_2,0)]}{W}, \tag{93}$$

$$r_f = \frac{\mathbb{E}[\mathcal{A}(\hat{Q}_1,\hat{Q}_2,\hat{Q}_3)]}{W}. \tag{94}$$

Proof: Consider the window backwards from the canonical pool block. For example, consider the pool block of a fork race. The window W first meets the race phase (pool work $\hat{q}_3$, density a_3'), of which it holds $a_3'\min\{\hat{q}_3, W\}$, then the withholding phase ($\hat{q}_2$, density a_2'), then the first phase ($\hat{q}_1$, density a_1'), and then earlier cycles. The length of the window left for the withholding phase is $(W-\hat{q}_3)^+$, thus it holds $a_2'\min\{\hat{q}_2, (W-\hat{q}_3)^+\}$; the length left for the first phase is $(W-\hat{q}_2-\hat{q}_3)^+$, thus it holds $a_1'\min\{\hat{q}_1, (W-\hat{q}_2-\hat{q}_3)^+\}$; the length left for earlier cycles is $(W-\hat{q}_1-\hat{q}_2-\hat{q}_3)^+$. This is (90). The other canonical pool blocks, i.e., those of Cases 3, 4a and 4b-ii and the released fPoW of Case 4b-iii, are similar with $\hat{q}_3 = 0$ and $\hat{q}_2 \in \{0, \hat{T}, \hat{Q}_2\}$. Since the cycle ends at the first of independent exponential clocks, $\hat{Q}_1, \hat{Q}_2, \hat{Q}_3$ are independent of which case occurs, and taking expectations completes the proof. ■

Combining the result of Lemma 2 with Theorem 7, we can prove Theorem 1 with

$$r_1: \quad \mathbb{E}\big[a_1'\hat{Q}_1 - \bar{x}\hat{Q}_1\big] = (a_1'-\bar{x})a_1, \tag{95}$$

$$r_s: \quad \mathbb{E}\big[a_1'\hat{Q}_1 + a_2'\hat{T} - \bar{x}(\hat{Q}_1+\hat{T})\big] = (a_1'-\bar{x})a_1 + (a_2'-\bar{x})\hat{T}, \tag{96}$$

$$r_u: \quad \mathbb{E}\big[a_1'\hat{Q}_1 + a_2'\hat{Q}_2 - \bar{x}(\hat{Q}_1+\hat{Q}_2)\big] = (a_1'-\bar{x})a_1 + (a_2'-\bar{x})\nu_1, \tag{97}$$

$$r_f: \quad \mathbb{E}\big[a_1'\hat{Q}_1 + a_2'\hat{Q}_2 + a_3'\hat{Q}_3 - \bar{x}(\hat{Q}_1+\hat{Q}_2+\hat{Q}_3)\big] = (a_1'-\bar{x})a_1 + (a_2'-\bar{x})\nu_1 + (a_3'-\bar{x})a_3. \tag{98}$$

The closed-forms neglect the $o(1)$ term of (77), evaluated at $v = W-\hat{Q}_1-\hat{Q}_2-\hat{Q}_3$, together with the event $\hat{Q}_1+\hat{Q}_2+\hat{Q}_3 > W$. Since the pool blockwork of a cycle is a fraction of one unit on average and has exponentially decaying tails, both are negligible for windows W of a few units of blockwork. If W is comparable to the pool blockwork of a cycle, Theorem 7 must be used directly.

APPENDIX B
REWARD FRACTIONS FOR ROUND-BASED PROP

A. Model

Under proportional (PROP) reward sharing, the pool operator splits the reward of a pool block among the members in proportion to the blockwork they submitted during the *round* of that block. We define the rounds by the canonical pool blocks as follows:

(R1) The round boundaries are the instants at which the operator receives canonical pool blocks. The reward of a canonical pool block is split in proportion to the blockwork submitted since the previous round boundary.

(R2) A pool block that is orphaned, and a withheld fPoW that is discarded, is not a round boundary. The blockwork submitted before and after it belongs to the same round.

A round is therefore not tied to an attack cycle and ends only with a canonical block of the pool. In particular, if the adversary releases a withheld fPoW to start a fork race, the release time is a round boundary if and only if the released fPoW wins the race to become canonical.

Remark 10 *A pool could alternatively close a round whenever it receives a fPoW, including a released fPoW that later loses the fork race. We believe our definition above is more likely to be used in practice. It is straightforward (and easier) to modify the analysis given here to obtain the reward fractions for the alternative case.*

The adversarial fraction of the pool blockwork is a'_1, a'_2, a'_3 in the first phase, the withholding phase and the fork race phase, respectively, and we refer to these regimes as modes $1, 2, 3$. If a round consists of multiple attack cycles, with total pool blockwork y_i submitted in mode i during the entire round, the adversarial fraction of the pool blockwork is

$$\varrho(\mathbf{y}) = \frac{a'_1 y_1 + a'_2 y_2 + a'_3 y_3}{y_1 + y_2 + y_3} \tag{99}$$

$$= a'_1 + \frac{(a'_2 - a'_1) y_2}{y_1 + y_2 + y_3} + \frac{(a'_3 - a'_1) y_3}{y_1 + y_2 + y_3}. \tag{100}$$

B. The Carry Process

The attack cycles remain i.i.d., but the share rewards of the pool do not: the reward split of a pool block depends on the blockwork of its round, which may span multiple attack cycles. Let us index the attack cycles by $n = 0, 1, 2, \ldots$ and let $\mathbf{C}^{(n)} = (C_1^{(n)}, C_2^{(n)}, C_3^{(n)})$ be the *carry*, i.e., the pool blockwork of the open round submitted in each mode, observed at the start of cycle n. Let ζ_n collect the randomness of cycle n. Define

$$\varepsilon = \alpha p_1 \pi_R (1 - \gamma) \xi, \tag{101}$$

$$\eta_3 = \alpha p_1 (1 - \theta) \pi_R (\alpha q_3 + \gamma \xi), \tag{102}$$

$$\eta_0 = \beta + \alpha p_1 \big(\theta + (1 - \theta)(\pi_P + \pi_R a_3)\big). \tag{103}$$

We consider how an attack cycle n that starts with carry $\mathbf{C}^{(n)} = \mathbf{C}$ evolves and list the probability and the blockwork associated with the canonical pool blocks mined during that cycle as well as the carry $\mathbf{C}^{(n+1)} = \mathbf{C}^+$ left to the next cycle:

Case 1, w.p. ξ no pool block. $\mathbf{C}^+ = \mathbf{C} + (\hat{Q}_1, 0, 0)$.

Case 2, w.p. αq_1 no pool block. $\mathbf{C}^+ = \mathbf{C} + (\hat{Q}_1, 0, 0)$.

Case 3, w.p. β: a canonical pool block closes the round with blockwork $\mathbf{C} + (\hat{Q}_1, 0, 0)$. $\mathbf{C}^+ = 0$.

Case 4a, w.p. $\alpha p_1 \theta$: the fPoW released is canonical and closes the round with blockwork $\mathbf{C} + (\hat{Q}_1, \hat{T}, 0)$. $\mathbf{C}^+ = 0$.

Case 4b-i, w.p. $\alpha p_1 (1 - \theta) \pi_A$: the fPoW is discarded, no canonical pool block. $\mathbf{C}^+ = \mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0)$.

Case 4b-ii, w.p. $\alpha p_1 (1 - \theta) \pi_P$: a canonical pool block closes the round with blockwork $\mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0)$. $\mathbf{C}^+ = 0$.

Case 4b-iii, w.p. $\alpha p_1 (1 - \theta) \pi_R$: the fPoW is released and the race is decided by next block.

Case 4b-iii-A, w.p. αq_3: the round closes with blockwork $\mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0)$ and $\mathbf{C}^+ = (0, 0, \hat{Q}_3)$.

Case 4b-iii-B, w.p. a_3:

$\mathbb{1}_{P_R} = 1$: the round closes with blockwork $\mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0)$ and the subsequent round is also closed with the canonical pool block of the race which has blockwork $(0, 0, \hat{Q}_3)$. $\mathbf{C}^+ = 0$.

$\mathbb{1}_{P_R} = 0$: the fPoW is orphaned, but the pool block of the race is canonical and closes the round with blockwork $\mathbf{C} + (\hat{Q}_1, \hat{Q}_2, \hat{Q}_3)$. $\mathbf{C}^+ = 0$.

Case 4b-iii-C, w.p. ξ:

w.p. γ: the round closes with blockwork $\mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0)$. $\mathbf{C}^+ = (0, 0, \hat{Q}_3)$.
w.p. $(1-\gamma)$: the fPoW is orphaned. $\mathbf{C}^+ = \mathbf{C} + (\hat{Q}_1, \hat{Q}_2, \hat{Q}_3)$.

Consequently,

$$\mathbf{C}^+ = \begin{cases} \mathbf{C} + (\hat{Q}_1, 0, 0), & \text{w.p. } \xi + \alpha q_1, \\ \mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0), & \text{w.p. } \alpha p_1 (1-\theta) \pi_A, \\ \mathbf{C} + (\hat{Q}_1, \hat{Q}_2, \hat{Q}_3), & \text{w.p. } (1-\theta)\varepsilon, \\ (0, 0, \hat{Q}_3), & \text{w.p. } \eta_3, \\ (0, 0, 0), & \text{w.p. } \eta_0, \end{cases} \tag{104}$$

and the five probabilities sum to one.

Note that $\mathbf{C}^{(n+1)}$ is a deterministic function of $(\mathbf{C}^{(n)}, \zeta_n)$, thus the carry is a Markov chain. We call cycle n a *reset cycle* if it is one of Cases 3, 4a, 4b-ii, 4b-iii-B. Whether a cycle is a reset cycle is a function of ζ_n only, it has probability $\eta_0 \geq \beta > 0$, and $\mathbf{C}^{(n+1)} = 0$ iff cycle n is a reset cycle. Notice this Markov chain is positive recurrent and regenerative, and by the renewal reward theorem

$$\rho_x = \frac{\mathbb{E}_\pi[B_x]}{\mathbb{E}[B_C]}, \qquad x \in \{A, P, R\}, \tag{105}$$

where $\mathbb{E}_\pi$ is taken with the carry in its stationary distribution.

Thus, the reward fractions are the expected adversarial fractions of the paid rounds, with $\mathbf{C} \sim \pi$ independent of $\hat{Q}_1, \hat{Q}_2, \hat{Q}_3$:

$$r_1 = \mathbb{E}_\pi\big[\varrho\big(\mathbf{C} + (\hat{Q}_1, 0, 0)\big)\big], \tag{106}$$

$$r_s = \mathbb{E}_\pi\big[\varrho\big(\mathbf{C} + (\hat{Q}_1, \hat{T}, 0)\big)\big], \tag{107}$$

$$r_u = \mathbb{E}_\pi\big[\varrho\big(\mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0)\big)\big], \tag{108}$$

$$r_m = \mathbb{E}_\pi\big[\varrho\big(\mathbf{C} + (\hat{Q}_1, \hat{Q}_2, \hat{Q}_3)\big)\big], \tag{109}$$

$$r_f = \mathbb{1}_{P_R}\, \mathbb{E}\big[\varrho\big((0, 0, \hat{Q}_3)\big)\big] + \overline{\mathbb{1}_{P_R}}\, r_m. \tag{110}$$

C. Transform of the Carry

Note that, for $s \geq 0$, we have

$$\mathbb{E}\big[e^{-s\hat{Q}_1}\big] = \phi_1(s) = \frac{1}{1 + a_1 s}, \tag{111}$$

$$\mathbb{E}\big[e^{-s\hat{Q}_3}\big] = \phi_3(s) = \frac{1}{1 + a_3 s}, \tag{112}$$

$$(1-\theta)\, \mathbb{E}\big[e^{-s\hat{Q}_2}\big] = \phi_2(s) = \frac{\mu_2\big(1 - \theta e^{-s\hat{T}}\big)}{\mu_2 + s}, \tag{113}$$

$$(1-\theta)\, \mathbb{E}\big[\hat{Q}_2 e^{-s\hat{Q}_2}\big] = \hat{\phi}_2(s) = -\phi_2'(s) = \frac{\mu_2\Big(1 - \theta e^{-s\hat{T}}\big(1 + (\mu_2 + s)\hat{T}\big)\Big)}{(\mu_2 + s)^2}. \tag{114}$$

Moreover, $\phi_2(0) = 1 - \theta$, $\hat{\phi}_2(0) = (1-\theta)\nu_1$, $\hat{\phi}_2 \geq 0$, ϕ_2 is non-increasing, $-\phi_3'(s) = a_3 \phi_3(s)^2$, and

$$\phi_2(s) \leq \frac{\mu_2}{\mu_2 + s} \leq 1, \qquad \hat{\phi}_2(s) \leq \frac{\mu_2}{(\mu_2 + s)^2}. \tag{115}$$

Lemma 3 *The stationary carry has the Laplace transform*

$$\Lambda(s_1, s_2, s_3) = \mathbb{E}_\pi\big[e^{-s_1 C_1 - s_2 C_2 - s_3 C_3}\big] = \frac{N(s_3)}{D(s_1, s_2, s_3)}, \tag{116}$$

where

$$N(s) = \eta_0 + \eta_3\phi_3(s), \tag{117}$$

$$D(s_1,s_2,s_3) = 1 - \phi_1(s_1)\Big(\xi + \alpha q_1 + \phi_2(s_2)\big(\alpha p_1\pi_A + \varepsilon\phi_3(s_3)\big)\Big). \tag{118}$$

Proof: Taking the transform of (104) in stationarity and using the independence of $\mathbf{C}$, $\hat{Q}_1$, $\hat{Q}_2$, $\hat{Q}_3$ gives $\Lambda = \eta_0 + \eta_3\phi_3(s_3) + \Lambda\big(1 - D(s_1,s_2,s_3)\big)$. ■

Lemma 4 *Let $\mathbf{Y} = (Y_1, Y_2, Y_3)$ be a non-negative random vector with $Y_1+Y_2+Y_3 > 0$ a.s. and Laplace transform $M(s_1,s_2,s_3)$. Then,*

$$\mathbb{E}\left[\frac{Y_i}{Y_1+Y_2+Y_3}\right] = -\int_0^\infty \frac{\partial M}{\partial s_i}(s,s,s)\,ds. \tag{119}$$

Proof: Use $\frac{1}{y} = \int_0^\infty e^{-sy}ds$ and Tonelli's theorem. ■

Lemma 5 (Transforms of the paid rounds) *With $\mathbf{C} \sim \pi$ independent of $\hat{Q}_1, \hat{Q}_2, \hat{Q}_3$, the Laplace transforms of the rounds in* (106)–(109) *are*

$$M_1 = \Lambda\,\phi_1(s_1), \qquad \text{for } \mathbf{C} + (\hat{Q}_1, 0, 0), \tag{120}$$

$$M_s = \Lambda\,\phi_1(s_1)\,e^{-s_2\hat{T}}, \qquad \text{for } \mathbf{C} + (\hat{Q}_1, \hat{T}, 0), \tag{121}$$

$$M_u = \Lambda\,\phi_1(s_1)\,\frac{\phi_2(s_2)}{1-\theta}, \qquad \text{for } \mathbf{C} + (\hat{Q}_1, \hat{Q}_2, 0), \tag{122}$$

$$M_m = \Lambda\,\phi_1(s_1)\,\frac{\phi_2(s_2)\phi_3(s_3)}{1-\theta}, \qquad \text{for } \mathbf{C} + (\hat{Q}_1, \hat{Q}_2, \hat{Q}_3), \tag{123}$$

with $\Lambda = \Lambda(s_1,s_2,s_3)$, and each of these rounds is non-zero a.s.

On the diagonal $s_1 = s_2 = s_3 = s$ we write $\phi_i = \phi_i(s)$, $\hat{\phi}_2 = \hat{\phi}_2(s)$, $N = N(s)$, $D = D(s,s,s)$, and define

$$K(s) = \frac{N\phi_1}{D}, \tag{124}$$

$$K_2(s) = \big(\alpha p_1\pi_A + \varepsilon\phi_3\big)\,\hat{\phi}_2\,\frac{\phi_1 K(s)}{D}, \tag{125}$$

$$K_3(s) = a_3\phi_3^2\,\frac{\phi_1}{D}\big(\eta_3 + \varepsilon\phi_2 K(s)\big). \tag{126}$$

Since $\Lambda\phi_1 = K$, differentiating N/D in Lemma 3 and using $-\phi_2' = \hat{\phi}_2$ and $-\phi_3' = a_3\phi_3^2$ gives, on the diagonal,

$$-\frac{\partial(\Lambda\phi_1)}{\partial s_2} = K_2(s), \qquad -\frac{\partial(\Lambda\phi_1)}{\partial s_3} = K_3(s). \tag{127}$$

By (100) and Lemma 4, $r_x = a_1' + (a_2' - a_1')\,\omega_2^x + (a_3' - a_1')\,\omega_3^x$ with $\omega_i^x = -\int_0^\infty \frac{\partial M_x}{\partial s_i}(s,s,s)\,ds$, the expected fraction of the blockwork of the round submitted in mode i. Applying the product rule to the transforms of Lemma 5 with (127), gives

$$\omega_2^1 = \int_0^\infty K_2(s)\,ds, \tag{128}$$

$$\omega_3^1 = \int_0^\infty K_3(s)\,ds, \tag{129}$$

$$\omega_2^s = \int_0^\infty e^{-s\hat{T}}\big(\hat{T}K(s) + K_2(s)\big)ds, \tag{130}$$

$$\omega_3^s = \int_0^\infty e^{-s\hat{T}}K_3(s)\,ds, \tag{131}$$

$$\omega_2^u = \frac{1}{1-\theta}\int_0^\infty \big(\hat{\phi}_2 K(s) + \phi_2 K_2(s)\big)ds, \tag{132}$$

$$\omega_3^u = \frac{1}{1-\theta}\int_0^\infty \phi_2 K_3(s)\,ds, \tag{133}$$

$$\omega_2^m = \frac{1}{1-\theta}\int_0^\infty \phi_3\big(\hat{\phi}_2 K(s) + \phi_2 K_2(s)\big)ds, \tag{134}$$

$$\omega_3^m = \frac{1}{1-\theta}\int_0^\infty \phi_2\phi_3\big(a_3\phi_3 K(s) + K_3(s)\big)ds. \tag{135}$$

Finally, $\mathbb{E}[\varrho((0,0,\hat{Q}_3))] = a_3'$ in (110), which concludes the proof of Theorem 2.

Remark 11 (GRP) *Under GRP, every block that becomes known to the pool operator starts a new round, and every attack cycle starts right after such a block. Hence, the open round is empty at the start of every cycle, i.e.,* $\mathbf{C} \equiv 0$ *and* $\Lambda \equiv 1$ *in Lemma 3. A withheld fPoW is not known to the operator, thus the blockwork of the withholding phase stays in the round, and the rounds of the pool blocks of Cases 3, 4a and 4b-ii, and of the released fPoW of Case 4b-iii, are* $(\hat{Q}_1, 0, 0)$, $(\hat{Q}_1, \hat{T}, 0)$ *and* $(\hat{Q}_1, \hat{Q}_2, 0)$ *as in* (106)–(108). *The honest block that triggers the fork race starts a new round, thus the round of the pool block of the race is* $(0, 0, \hat{Q}_3)$ *regardless of* $\mathbb{1}_{P_R}$, *which gives* (29). *With* $\Lambda \equiv 1$, *Lemma 5 gives* $M_1 = \phi_1(s_1)$, $M_s = \phi_1(s_1)e^{-s_2\hat{T}}$ *and* $M_u = \phi_1(s_1)\phi_2(s_2)/(1-\theta)$, *thus* $\omega_3^x = 0$, $\omega_2^1 = 0$, *and by Lemma 4*

$$\omega_2^s = \int_0^\infty \phi_1\,\hat{T}e^{-s\hat{T}}ds, \qquad \omega_2^u = \frac{1}{1-\theta}\int_0^\infty \phi_1\hat{\phi}_2\,ds.$$

For $y \geq 0$, *the substitution* $t = 1 + a_1 s$ *gives*

$$\int_0^\infty \frac{ye^{-sy}}{1+a_1 s}\,ds = \frac{y}{a_1}e^{y/a_1}E_1\Big(\frac{y}{a_1}\Big),$$

which gives (27) *with* $y = \hat{T}$. *Since* $\hat{\phi}_2(s) = (1-\theta)\mathbb{E}[\hat{Q}_2 e^{-s\hat{Q}_2}]$ *and* $\hat{Q}_2$ *has the density* $\frac{\mu_2 e^{-\mu_2 y}}{1-\theta}$ *on* $[0,\hat{T}]$, *the same identity gives* (28).

APPENDIX C
REWARD FRACTIONS FOR EXP

A. Model

The rounds are defined by the canonical pool blocks as in (R1)–(R2) of Appendix B, and the attack cycle evolves through the same cases. Under EXP scheme with decay rate λ_d, a share submitted t time units before the end of its round receives the weight $w(t) = e^{-\lambda_d t}$, and the reward of a canonical pool block is split in proportion to the total weights of the shares of its round. As before, the pool submits blockwork at rate $\lambda_1 a_i$ in mode i, of which $\lambda_1 \alpha p_i$ is adversarial. For a round that closes at time t_c, we define its *score* Z and *adversarial score* Z_A as

$$Z = \int_{\text{round}} \lambda_1 a(t)\, e^{-\lambda_d(t_c - t)}\,dt, \tag{136}$$

$$Z_A = \int_{\text{round}} \lambda_1 \alpha p(t)\, e^{-\lambda_d(t_c - t)}\,dt, \tag{137}$$

where $a(t) = a_i$ and $p(t) = p_i$ if t belongs to mode i, and we write $\mathbf{Z} = (Z_A, Z)$. The adversarial fraction of the reward of the round is Z_A/Z. We use

$$\kappa_1 = \frac{\lambda_1}{\lambda_d}, \quad \kappa_2 = \frac{\lambda_2}{\lambda_d}, \quad \chi_T = e^{-\lambda_d T}, \quad \mathbf{a}_i = (\alpha p_i, a_i), \tag{138}$$

so that $\theta = \chi_T^{\kappa_2}$. As in Appendix A, the first phase and the fork race last $T_1, T_3 \sim \text{Exp}(\lambda_1)$, and a withholding phase that ends before the deadline (Case 4b, probability $1-\theta$) lasts $T_2' \sim (T_2 \mid T_2 < T)$ with $T_2 \sim \text{Exp}(\lambda_2)$; a withholding phase that reaches the deadline (Case 4a) lasts exactly T. We use the corresponding discount factors

$$\chi_1 = e^{-\lambda_d T_1}, \quad \chi_2' = e^{-\lambda_d T_2'}, \quad \chi_3 = e^{-\lambda_d T_3}, \tag{139}$$

and χ_T. Since $\mathbb{P}(\chi_1 \le u) = \mathbb{P}(T_1 \ge -\ln u/\lambda_d) = u^{\kappa_1}$, χ_1 and χ_3 have the density $\kappa_1 u^{\kappa_1 - 1}$ on $[0,1]$. Similarly, $\{T_2 < T\} = \{e^{-\lambda_d T_2} > \chi_T\}$, thus χ_2' has the density

$$\frac{\kappa_2 u^{\kappa_2 - 1}}{1-\theta}\, \mathbb{1}_{\{u > \chi_T\}}, \qquad u \in [0,1]. \tag{140}$$

Lemma 6 (Score update) *If the pool mines in mode i for a duration t without closing the round, the score of the open round changes as*

$$\mathbf{Z} \;\mapsto\; e^{-\lambda_d t}\,\mathbf{Z} + \kappa_1\big(1 - e^{-\lambda_d t}\big)\mathbf{a}_i. \tag{141}$$

Proof: Moving the evaluation instant forward by t multiplies the weight of every earlier share by $e^{-\lambda_d t}$, and the new shares contribute $\int_0^t \lambda_1 a_i e^{-\lambda_d(t-t')}dt' = \kappa_1(1 - e^{-\lambda_d t})a_i$ to Z and $\kappa_1(1 - e^{-\lambda_d t})\alpha p_i$ to Z_A. ■

Hence, the score updates over the first phase, the withholding phase ending before the deadline, the withholding phase ending at the deadline and the fork race are the affine maps

$$\mathcal{U}_1(\mathbf{z}) = \chi_1 \mathbf{z} + \kappa_1(1-\chi_1)\mathbf{a}_1, \tag{142}$$
$$\mathcal{U}_2(\mathbf{z}) = \chi_2' \mathbf{z} + \kappa_1(1-\chi_2')\mathbf{a}_2, \tag{143}$$
$$\mathcal{U}_T(\mathbf{z}) = \chi_T \mathbf{z} + \kappa_1(1-\chi_T)\mathbf{a}_2, \tag{144}$$
$$\mathcal{U}_3(\mathbf{z}) = \chi_3 \mathbf{z} + \kappa_1(1-\chi_3)\mathbf{a}_3. \tag{145}$$

B. The Carry Process

Let $\mathbf{Z}^{(n)}$ be the *carry*, i.e., the score of the open round evaluated at the start of cycle n. For a cycle that starts with carry $\mathbf{Z}^{(n)} = \mathbf{Z}$ and for a score $\mathbf{z} = (z_A, z)$, let $\varrho_d(\mathbf{z}) = z_A/z$. Following the case-by-case analysis of Appendix B, the canonical pool blocks are paid with the scores $\mathcal{U}_1(\mathbf{Z})$ in Case 3, $\mathcal{U}_T(\mathcal{U}_1(\mathbf{Z}))$ in Case 4a, $\mathcal{U}_2(\mathcal{U}_1(\mathbf{Z}))$ in Case 4b-ii and for the canonical fPoW of Case 4b-iii, and, in Case 4b-iii-B, $\mathcal{U}_3(\mathbf{0})$ if $\mathbb{1}_{P_R} = 1$ and $\mathcal{U}_3(\mathcal{U}_2(\mathcal{U}_1(\mathbf{Z})))$ if $\mathbb{1}_{P_R} = 0$. The carry left to the next cycle is

$$\mathbf{Z}^{(n+1)} = \begin{cases} \mathcal{U}_1(\mathbf{Z}), & \text{w.p. } \xi + \alpha q_1, \\ \mathcal{U}_2(\mathcal{U}_1(\mathbf{Z})), & \text{w.p. } \alpha p_1 \pi_A (1-\theta), \\ \mathcal{U}_3(\mathcal{U}_2(\mathcal{U}_1(\mathbf{Z}))), & \text{w.p. } \varepsilon(1-\theta), \\ \mathcal{U}_3(\mathbf{0}), & \text{w.p. } \eta_3, \\ \mathbf{0}, & \text{w.p. } \eta_0, \end{cases} \tag{146}$$

with $\varepsilon, \eta_3, \eta_0$ in (101)–(103). That is, (146) is (104) with the blockwork of each phase replaced by the corresponding score update under exponential decay. In particular, the reset cycles are the same, and by the renewal reward theorem $\rho_x = \mathbb{E}_\pi[B_x]/\mathbb{E}[B_C]$ with the carry in its stationary distribution π. The reward fractions are

$$r_1 = \mathbb{E}_\pi\big[\varrho_d\big(\mathcal{U}_1(\mathbf{Z})\big)\big], \tag{147}$$
$$r_s = \mathbb{E}_\pi\big[\varrho_d\big(\mathcal{U}_T(\mathcal{U}_1(\mathbf{Z}))\big)\big], \tag{148}$$
$$r_u = \mathbb{E}_\pi\big[\varrho_d\big(\mathcal{U}_2(\mathcal{U}_1(\mathbf{Z}))\big)\big], \tag{149}$$
$$r_m = \mathbb{E}_\pi\big[\varrho_d\big(\mathcal{U}_3(\mathcal{U}_2(\mathcal{U}_1(\mathbf{Z})))\big)\big], \tag{150}$$

where $\mathbf{Z} \sim \pi$ is independent of χ_1, χ_2', χ_3, and $r_f = \mathbb{1}_{P_R} a_3' + \overline{\mathbb{1}_{P_R}}\, r_m$, since the fresh score $\mathcal{U}_3(\mathbf{0}) = \kappa_1(1-\chi_3)\mathbf{a}_3$ has the fraction a_3'.

C. Transform of the Carry

For a random score $\mathbf{Z} = (Z_A, Z)$, let

$$\mathcal{L}(s) = \mathbb{E}\big[e^{-sZ}\big], \quad \mathcal{L}_A(s) = \mathbb{E}\big[Z_A e^{-sZ}\big], \quad s \ge 0. \tag{151}$$

For a mode i and a measure $\mathfrak{m}$ on $[0,1]$, define the operator $\mathcal{G}_i^{\mathfrak{m}}(\mathcal{L}, \mathcal{L}_A) = (\mathcal{L}', \mathcal{L}_A')$ by

$$\mathcal{L}'(s) = \int_{[0,1]} e^{-s\kappa_1 a_i (1-u)}\, \mathcal{L}(su)\, \mathfrak{m}(du), \tag{152}$$

$$\mathcal{L}'_A(s) = \int_{[0,1]} e^{-s\kappa_1 a_i(1-u)} \Big(\kappa_1 \alpha p_i (1-u)\, \mathcal{L}(su) + u\, \mathcal{L}_A(su) \Big)\, \mathfrak{m}(du), \tag{153}$$

and write $\mathcal{G}_1$ and $\mathcal{G}_3$ for the law of χ_1 (and χ_3), $\mathfrak{m}(du) = \kappa_1 u^{\kappa_1 - 1} du$, with $i = 1$ and $i = 3$, $\mathcal{G}_2$ for the law (140) of χ'_2 with $i = 2$, and $\mathcal{G}_T$ for the point mass at χ_T with $i = 2$.

Lemma 7 *Let* $\mathbf{Z}$ *be a random score with transforms* $(\mathcal{L}, \mathcal{L}_A)$*, and* χ *a random variable on* $[0,1]$ *with law* $\mathfrak{m}$*, independent of* $\mathbf{Z}$*. Then, the transforms of* $\chi\mathbf{Z} + \kappa_1(1-\chi)\mathbf{a}_i$ *are* $\mathcal{G}_i^{\mathfrak{m}}(\mathcal{L}, \mathcal{L}_A)$*.*

Proof: Conditioning on $\chi = u$, the new score is $(uZ_A + \kappa_1 \alpha p_i(1-u),\ uZ + \kappa_1 a_i(1-u))$, thus $\mathbb{E}[e^{-sZ'} \mid \chi = u] = e^{-s\kappa_1 a_i(1-u)} \mathcal{L}(su)$ and $\mathbb{E}[Z'_A e^{-sZ'} \mid \chi = u] = e^{-s\kappa_1 a_i(1-u)} \big(\kappa_1 \alpha p_i (1-u) \mathcal{L}(su) + u \mathcal{L}_A(su) \big)$. Integrating against $\mathfrak{m}$ gives (152)–(153). ■

Theorem 8 *The transforms* $(\mathcal{L}, \mathcal{L}_A)$ *of the stationary carry are the unique bounded solution of*

$$(\mathcal{L}, \mathcal{L}_A) = \eta_0\, (1,0) + \eta_3\, \mathcal{G}_3(1,0) + (\xi + \alpha q_1)\, \mathcal{G}_1(\mathcal{L}, \mathcal{L}_A) + \alpha p_1 \pi_A (1-\theta)\, \mathcal{G}_2 \mathcal{G}_1(\mathcal{L}, \mathcal{L}_A) + \varepsilon(1-\theta)\, \mathcal{G}_3 \mathcal{G}_2 \mathcal{G}_1(\mathcal{L}, \mathcal{L}_A), \tag{154}$$

where $(1,0)$ *is the pair of transforms of the zero carry. With*

$$(\mathcal{L}^1, \mathcal{L}^1_A) = \mathcal{G}_1(\mathcal{L}, \mathcal{L}_A), \tag{155}$$

$$(\mathcal{L}^s, \mathcal{L}^s_A) = \mathcal{G}_T \mathcal{G}_1(\mathcal{L}, \mathcal{L}_A), \tag{156}$$

$$(\mathcal{L}^u, \mathcal{L}^u_A) = \mathcal{G}_2 \mathcal{G}_1(\mathcal{L}, \mathcal{L}_A), \tag{157}$$

$$(\mathcal{L}^m, \mathcal{L}^m_A) = \mathcal{G}_3 \mathcal{G}_2 \mathcal{G}_1(\mathcal{L}, \mathcal{L}_A), \tag{158}$$

the expected adversarial reward fractions under EXP scheme are

$$r_x = \int_0^\infty \mathcal{L}^x_A(s)\, ds, \qquad x \in \{1, s, u, m\}, \tag{159}$$

$$r_f = \mathbb{1}_{P_R}\, a'_3 + \overline{\mathbb{1}_{P_R}}\, r_m. \tag{160}$$

Proof: Taking the transforms (151) of both sides of (146) in stationarity, and applying Lemma 7 once per phase, in the order in which the phases occur, gives (154). The first component of (154) does not involve $\mathcal{L}_A$, and the second is affine in $\mathcal{L}_A$ for a given $\mathcal{L}$. Since $e^{-s\kappa_1 a_i(1-u)} \le 1$ and $u \le 1$, both are contractions in the supremum norm with factor $\xi + \alpha q_1 + (1-\theta)(\alpha p_1 \pi_A + \varepsilon) = 1 - \eta_0 - \eta_3 < 1$, hence the solution is unique and the iteration of (154) converges to it. By Lemma 7, $(\mathcal{L}^x, \mathcal{L}^x_A)$ are the transforms of the paid scores. Finally, for a paid score $\mathbf{Z}$, use $\mathbb{E}[\varrho_d(\mathbf{Z})] = \int_0^\infty \mathbb{E}[Z_A e^{-sZ}] ds$ by $\frac{1}{z} = \int_0^\infty e^{-sz} ds$ and Tonelli's theorem to get the reward fractions. ■

Every operator in (154) is a one-dimensional integral over $u \in [0,1]$ that evaluates $\mathcal{L}$ and $\mathcal{L}_A$ at $su \le s$, thus (154) can be solved on a grid in s by iteration, and r_x follows by one-dimensional quadrature.

APPENDIX D
PRACTICAL MINING POOLS: VARIABLE SHARE DIFFICULTY

If the mining pool employs vardiff to maintain an approximately constant share submission rate, the share difficulty d_p assigned to the adversary remains fixed because the adversary's fractional hashrate directed toward the target pool is assumed to remain constant at p, even as the fPoW difficulty changes across epochs. Furthermore, suppose that in the first epoch, the fPoW difficulty is d_f with $d_b = \frac{d_p}{d_f}$ denoting its first-epoch value. Note that, under vardiff,

$$\kappa_s \triangleq \frac{\lambda_1}{\lambda_{\text{share}}}, \tag{161}$$

which is fixed by the pool configuration. Since the pool assigns a share difficulty such that the infiltrating adversary (controlling αp fraction of the total network hashpower) submits one share every $1/\lambda_{\text{share}}$ seconds on average, we have

$$d_b = \alpha p \kappa_s. \tag{162}$$

Let $R_{s,j}$ denote the PPS reward per share during epoch j. During the first epoch, the reward per share is $R_{s,1} = R_b d_b$. Since the first DA occurs after the first epoch, the results of Lemma 1 also hold in this setting. After the first DA, the share difficulty for adversarial pool mining remains d_p, since the target share rate is fixed to λ_{share} and the adversary continues to allocate a fraction p of its hashpower toward the target pool. Meanwhile, due to the increased duration of the first epoch, the fPoW difficulty decreases to

$$d_{f,2} = \frac{d_f}{\delta^{(p)}} \tag{163}$$

for the second epoch and subsequent epochs. Consequently, after the first epoch, the reward per share increases to

$$R_{s,j} = \delta^{(p)} R_{s,1}, \quad j \geq 2. \tag{164}$$

This leads to the following result.

Corollary 3 *Under the PPS scheme (with vardiff), if the adversary initiates a classical BWH against a target pool with p fraction of its power from $t = 0$ onwards, after the first epoch, the revenue change of each mining entity is given by,*

$$\Delta_A^{BWH,H}(t_1 + t) = \frac{\alpha t}{\tau_0}\big(\delta^{(p)}(1 - p\delta^{(p)} d_b) - 1\big) - \alpha\delta^{(p)} p d_b \approx \frac{\alpha t}{\tau_0}(\delta^{(p)} - 1), \tag{165}$$

$$\Delta_P^{BWH,H}(t_1 + t) = \frac{\beta t}{\tau_0}(\delta^{(p)} - 1), \tag{166}$$

$$\Delta_R^{BWH,H}(t_1 + t) = \frac{\xi t}{\tau_0}(\delta^{(p)} - 1), \tag{167}$$

whereas the revenue change of the pool operator is,

$$\Delta_{P_o}^{BWH,H}(t_1 + t) = -\delta^{(p)}\alpha p(1 - d_b) - \Big(\frac{\alpha t}{\tau_0}\Big)\delta^{(p)} p(1 - \delta^{(p)} d_b). \tag{168}$$

Proof: The only difference compared with Corollary 1 is the term $(1 - p\delta^{(p)} d_b)$, which replaces $(1 - p d_b)$ in both the adversarial revenue change and the pool operator's revenue change. This difference arises because, under vardiff, the pPoW difficulty assigned to the adversary remains fixed, while the fPoW difficulty decreases after the first DA. As a result, after the first DA, the probability that a share is also a valid fPoW increases from d_b to $\delta^{(p)} d_b$, because the pPoW difficulty remains fixed while the fPoW difficulty decreases. Consequently, the reward per share increases to $R_{s,j} = \delta^{(p)} R_{s,1}$, $j \geq 2$.

Therefore, in each subsequent epoch, the adversary, which contributes a fraction p of its total hashpower to the target pool, receives

$$P_{j,A_P} = \delta^{(p)}\alpha p(1 - \delta^{(p)} d_b), \quad j \geq 2, \tag{169}$$

in PPS rewards over the epoch. ■

Finally, to prove Theorem 5, note that, after the first DA, the rate of change of adversarial revenue (its slope) is

$$\frac{\alpha}{\tau_0}\big(\delta^{(p)}(1 - p\delta^{(p)} d_b) - 1\big). \tag{170}$$

Our goal is to maximize this slope, i.e., the objective can be written as

$$f(p) = \delta^{(p)}\left(1 - p\delta^{(p)} d_b\right) - 1 \tag{171}$$

$$= \frac{1}{1 - \alpha p}\left(1 - \frac{\alpha\kappa_s p^2}{1 - \alpha p}\right) - 1 \tag{172}$$

$$= \frac{\alpha p\,(1 - (\alpha + \kappa_s)p)}{(1 - \alpha p)^2}. \tag{173}$$

Differentiating with respect to p yields

$$f'(p) = \frac{-\alpha\,((\alpha + 2\kappa_s)p - 1)}{(1 - \alpha p)^3}. \tag{174}$$

Since $0 \le \alpha p < 1$, we have $(1-\alpha p)^3 > 0$. Hence,

$$f'(p) \begin{cases} > 0, & p < \dfrac{1}{\alpha + 2\kappa_s}, \\ < 0, & p > \dfrac{1}{\alpha + 2\kappa_s}, \end{cases} \tag{175}$$

which implies that $f(p)$ is increasing for $p < (\alpha + 2\kappa_s)^{-1}$ and decreasing for $p > (\alpha + 2\kappa_s)^{-1}$. Therefore,

$$p^* = \begin{cases} 1, & \alpha + 2\kappa_s < 1, \\ \dfrac{1}{\alpha + 2\kappa_s}, & \alpha + 2\kappa_s \ge 1. \end{cases} \tag{176}$$

The corresponding maximum value is

$$f(p^*) = \begin{cases} \dfrac{\alpha(1-\alpha-\kappa_s)}{(1-\alpha)^2}, & \alpha + 2\kappa_s < 1, \\ \dfrac{\alpha}{4\kappa_s}, & \alpha + 2\kappa_s \ge 1. \end{cases} \tag{177}$$

Next, we note that κ_s can be made arbitrarily small by employing multiple sybil identities when joining the pool. Specifically, suppose the adversary splits its infiltration hashpower αp equally among m sybil identities. Since the pool independently targets a share submission rate of λ_{share} for each identity, the aggregate share submission rate becomes $m\lambda_{\text{share}}$, and the effective parameter is reduced to $\kappa_{s,\text{eff}} = \frac{\kappa_s}{m}$. As $m \to \infty$, we have $\kappa_{s,\text{eff}} \to 0$. Since $\alpha < 0.5$ by assumption, there always exists a sufficiently large m such that $\alpha + 2\kappa_{s,\text{eff}} < 1$. Therefore, the All-out Attack is optimal.

Appendix E
Revenue Ratios of Pool Hopping

Lemma 8 *Under PAH, the rounds are i.i.d. Let τ, $y(t)$ in* (55) *and $Y_\tau = y(\tau)$ be as defined in Section VII. Then,*

(i) $\mathbb{P}(\tau > t) = \bar{F}_\tau(t)$ *with* $\bar{F}_\tau(t) = e^{-\lambda_1 a_y t}$ *for* $t < t^*$ *and* $\bar{F}_\tau(t) = e^{*} e^{-\lambda_1 a_o (t - t^*)}$ *for* $t \ge t^*$*, and the pool block is found by the adversary with probability* $a'(\tau)$*;*

(ii) the expected numbers of honest blocks outside the pool and solo adversarial blocks found before τ are n_R and n_S;

(iii) $Y_\tau \sim \mathrm{Exp}(1)$ *for every policy* (p_y, p_o, t^*)*.*

Proof: A round starts at a stopping time of the mining processes and the policy depends only on the time since the start of the round, hence the rounds are i.i.d. by the memoryless property. At time t, the pool mines with the hash fraction $a(t)$, hence the pool blocks arrive with the intensity $\lambda_1 a(t)$, which gives $\bar{F}_\tau$. A pool block at time t is adversarial with probability $\frac{\alpha p(t)}{a(t)} = a'(t)$. The outside and solo blocks arrive with the intensities $\lambda_1 \xi$ and $\lambda_1 \alpha q(t)$ and do not end the round, thus their expected numbers are $\int_0^\infty \bar{F}_\tau(t) \lambda_1 \xi \, dt$ and $\int_0^\infty \bar{F}_\tau(t) \lambda_1 \alpha q(t) \, dt$, which give (59) and (60).

For (iii), the idea is to measure the round by the work done in it instead of by time. Every hash is a valid block with the same probability, regardless of who computes it and how fast. Hence, the probability that the pool finds a block depends only on how much blockwork the pool has done, and not on how long it took. The blockwork age counts exactly this work, thus every additional unit of blockwork age carries the same chance of ending the round, i.e., in units of blockwork age the round ends at the constant rate 1. The power allocation of the adversary only changes how fast the blockwork age grows, i.e., faster with p_y and slower with p_o, but not how much blockwork age a round lasts.

Formally, (55) is deterministic, continuous and strictly increasing since $a(t) \ge \beta > 0$, and $y(t^*) = y^*$. By (i),

$$\mathbb{P}(\tau > t) = \exp\Big(- \int_0^t \lambda_1 a(t') \, dt' \Big) = e^{-y(t)}, \tag{178}$$

and since $y(\cdot)$ is strictly increasing, $\{\tau > t\} = \{Y_\tau > y(t)\}$. Hence, $\mathbb{P}(Y_\tau > y) = e^{-y}$ for all $y \geq 0$, i.e., $Y_\tau \sim \mathrm{Exp}(1)$. Equivalently, the probability of a pool block in $[t, t+dt)$ is $\lambda_1 a(t)\, dt = dy(t)$, the increment of the blockwork age, hence the pool blocks form a Poisson process of unit rate in units of blockwork age, which is the time change of a Poisson process. ■

A. Proof of Theorem 6

Nothing is withheld, therefore every block is canonical and $\delta^{\mathrm{PAH}} = 1$. Every round contains exactly one pool block, which pays 1 in total, and by Lemma 8(ii), n_S solo and n_R outside blocks on average, each paying 1. By the renewal reward theorem, ρ_x is the expected reward of x per round divided by the expected number of canonical blocks per round, $n_S + n_R + 1$, which gives (56)-(58) with r_h the expected adversarial reward fraction of the pool block. It remains to compute r_h.

PROP: The pool block pays the adversarial fraction of the blockwork of the round. By Lemma 8(iii), $Y_\tau \sim \mathrm{Exp}(1)$, and the adversarial blockwork at blockwork age y is $b_A(y) = a'_y y$ for $y < y^*$ and $b_A(y) = a'_o y + (a'_y - a'_o)y^*$ for $y \geq y^*$. Hence,

$$r_h^{\mathrm{PROP}} = \mathbb{E}\left[\frac{b_A(Y_\tau)}{Y_\tau}\right] = \int_0^{y^*} e^{-y} a'_y\, dy + \int_{y^*}^{\infty} e^{-y}\Big(a'_o + \frac{(a'_y - a'_o)y^*}{y}\Big) dy, \tag{179}$$

which is (61) since $\int_{y^*}^{\infty} \frac{e^{-y}}{y} dy = E_1(y^*)$.

EXP: With $a(t)$ and $p(t)$ given by the policy (52), the score and the adversarial score (136)–(137) of the open round evaluated at time t are deterministic,

$$Z(t) = \int_0^t \lambda_1 a(t') e^{-\lambda_d (t-t')} dt', \tag{180}$$

$$Z_A(t) = \int_0^t \lambda_1 \alpha p(t') e^{-\lambda_d (t-t')} dt', \tag{181}$$

and the pool block at time τ pays the adversary $Z_A(\tau)/Z(\tau)$. For $t < t^*$, the pool hashing is constant, thus $Z_A/Z = a'_y$, and the pool block falls before t^* with probability $1 - e^*$. For $t = t^* + u$ with $u \geq 0$, by Lemma 6,

$$\frac{Z(t)}{\kappa_1} = a_y(1-\chi^*)e^{-\lambda_d u} + a_o\big(1 - e^{-\lambda_d u}\big), \tag{182}$$

$$\frac{Z_A(t)}{\kappa_1} = \alpha p_y(1-\chi^*)e^{-\lambda_d u} + \alpha p_o\big(1 - e^{-\lambda_d u}\big), \tag{183}$$

and, given $\tau \geq t^*$, $\tau - t^* \sim \mathrm{Exp}(\lambda_1 a_o)$ by Lemma 8(i). Hence,

$$r_h^{\mathrm{EXP}} = a'_y(1-e^*) + e^* \int_0^{\infty} \lambda_1 a_o e^{-\lambda_1 a_o u}\, \frac{Z_A(t^*+u)}{Z(t^*+u)}\, du, \tag{184}$$

and the substitution $v = e^{-\lambda_d u}$, under which $\lambda_1 a_o e^{-\lambda_1 a_o u} du = -\kappa_o v^{\kappa_o - 1} dv$, gives (62). ■

B. All-or-Nothing Hopping

Let $p_y = 1$ and $p_o = 0$, i.e., the adversary mines only in the pool during the first t^* time units of a round, which correspond to the first $y^* = \lambda_1(\alpha+\beta)t^*$ blockworks, and only solo afterwards. Then, $a_y = \alpha + \beta$, $a_o = \beta$, $a'_y = \frac{\alpha}{\alpha+\beta}$, $a'_o = 0$, and

$$n_S = \frac{\alpha}{\beta} e^*, \qquad n_S + n_R + 1 = \frac{1-e^*}{\alpha+\beta} + \frac{e^*}{\beta}. \tag{185}$$

Corollary 4 *Let $\beta' = \frac{\beta}{\alpha+\beta}$. Under round-based PROP,*

$$\frac{\rho_A}{\alpha} - 1 = \frac{\beta'\, y^* E_1(y^*)}{\beta' + (1-\beta')e^*}. \tag{186}$$

Hence, in this case, the relative extra revenue depends on only α/β, and it is maximized at the root y^* of

$$\big(E_1(y^*) - e^{-y^*}\big)\big(\beta' + (1-\beta')e^{-y^*}\big) + (1-\beta')\,e^{-y^*}y^*E_1(y^*) = 0. \tag{187}$$

As $\alpha/\beta \to 0$, i.e., $\beta' \to 1$, (186) tends to $y^*E_1(y^*)$, which is maximized at the root $y^* \approx 0.4348$ of $e^{y^*}E_1(y^*) = 1$, where it equals $y^*e^{-y^*} \approx 0.2815$, which recovers the classical pool hopping result of [2], [21] for a negligible miner. For a non-negligible adversary, its own shares dilute the young rounds it mines in, which is captured by β' in (186)*; as a result, the gain decreases and the optimal threshold y^* increases with α/β.*

Proof: By (61), $r_h^{\mathrm{PROP}} = \frac{\alpha}{\alpha+\beta}\big(1 - e^* + y^*E_1(y^*)\big)$, hence by (56) and (185),

$$\frac{\rho_A}{\alpha} = \frac{\frac{e^*}{\beta} + \frac{1-e^*+y^*E_1(y^*)}{\alpha+\beta}}{\frac{e^*}{\beta} + \frac{1-e^*}{\alpha+\beta}} \tag{188}$$

$$= 1 + \frac{y^*E_1(y^*)/(\alpha+\beta)}{e^*/\beta + (1-e^*)/(\alpha+\beta)}, \tag{189}$$

and multiplying the numerator and the denominator of the last fraction by β gives (186). The derivative of $yE_1(y)$ is $E_1(y) - e^{-y}$ and that of $\beta' + (1-\beta')e^{-y}$ is $-(1-\beta')e^{-y}$, which gives (187). For $\beta' = 1$, (187) reduces to $E_1(y^*) = e^{-y^*}$. ■

Corollary 5 *Let $\varsigma = 1 - \frac{\alpha+\beta}{\beta}(1-\chi^*)$. Under EXP, with $\kappa_o = \lambda_1\beta/\lambda_d$,*

$$r_h^{\mathrm{EXP}} = \frac{\alpha}{\alpha+\beta}(1-e^*) + e^*\,\frac{\kappa_o}{\kappa_o+1}\,\frac{\alpha(1-\chi^*)}{\beta} \times {}_2F_1(1, \kappa_o+1; \kappa_o+2; \varsigma), \tag{190}$$

where ${}_2F_1$ is the Gaussian hypergeometric function.

Proof: With $p_o = 0$ the integrand of (62) is $v^{\kappa_o - 1} \cdot \frac{\alpha(1-\chi^*)v}{\beta(1-\varsigma v)}$, and $\int_0^1 \frac{v^{\kappa_o}}{1-\varsigma v}dv = \frac{1}{\kappa_o+1}\,{}_2F_1(1, \kappa_o+1; \kappa_o+2; \varsigma)$ by Euler's integral representation; note that $\varsigma < 1$ for $t^* > 0$. ■

Unlike (186), the gain under EXP depends on α and β separately, since the decay acts in time while the rounds end at the rate $\lambda_1 a(t)$, which depends on the size of the pool. If the rounds are long compared to $1/\lambda_d$, the score of an old round saturates, its composition is that of the recent shares only, and hopping gains little; if the rounds are short compared to $1/\lambda_d$, EXP behaves like PROP.

C. Pool Hopping Under W-PPLNS

The following is the hopping-proofness of PPLNS shown in [2, Sec. 3.3], which we state for an adversary of any size and any power allocation policy.

Proposition 1 (Rosenfeld [2]) *Under W-PPLNS with constant block rewards, every policy under which the adversary submits all its shares immediately and chooses its power allocation based only on the past of the mining processes gives $\rho_A = \alpha$.*

Proof: Consider a share of blockwork dv. It receives the fraction dv/W of the reward of every pool block found while it is among the last W blockworks of pool shares, i.e., during the next W blockworks of pool work. Measured by the pool blockwork, the pool blocks arrive as a Poisson process of unit rate whatever the allocation is, by the time change argument in the proof of Lemma 8(iii), and its future is independent of the past. Hence, the expected number of pool blocks during the next W blockworks is W, given anything that is known when the share is submitted, and the expected reward of the share is dv, which is the expected reward of the blockwork dv mined solo. Summing over all shares, the expected reward of the adversary per unit time equals that of mining solo with all its hashpower, which is α times the expected canonical reward per unit time, since all blocks are canonical. ■

Thus, a gain under W-PPLNS requires information about future pool blocks that is not contained in the past of the mining processes, such as a withheld fPoW whose release time the adversary controls, which is the idea behind lie-in-wait [2] and T-PAW. In other words, under W-PPLNS, any extra revenue of PAW/T-PAW over honest mining stems from withholding and not from power adjusting alone.

Attack	Scheme	β	γ	RER (%) Theory	RER (%) MC	δ Theory	δ MC
T-PAW	PPLNS	0.05	0	0.083	0.062 ± 0.092	1.0014	1.0015 ± 0.0002
T-PAW	PPLNS	0.05	0.5	0.195	0.167 ± 0.066	1.0032	1.0032 ± 0.0002
T-PAW	PPLNS	0.1	0	0.177	0.111 ± 0.046	1.0031	1.0031 ± 0.0002
T-PAW	PPLNS	0.1	0.5	0.376	0.375 ± 0.071	1.0062	1.0062 ± 0.0002
T-PAW	PROP	0.05	0	0.194	0.157 ± 0.085	1.0016	1.0015 ± 0.0003
T-PAW	PROP	0.05	0.5	0.544	0.511 ± 0.077	1.0046	1.0046 ± 0.0002
T-PAW	PROP	0.1	0	0.298	0.295 ± 0.080	1.0028	1.0029 ± 0.0002
T-PAW	PROP	0.1	0.5	0.724	0.667 ± 0.071	1.0075	1.0074 ± 0.0002
T-PAW	EXP	0.05	0	2.382	2.434 ± 0.053	1.0031	1.0032 ± 0.0002
T-PAW	EXP	0.05	0.5	3.425	3.403 ± 0.063	1.0054	1.0054 ± 0.0002
T-PAW	EXP	0.1	0	1.955	1.961 ± 0.078	1.0043	1.0043 ± 0.0002
T-PAW	EXP	0.1	0.5	2.800	2.805 ± 0.084	1.0072	1.0071 ± 0.0003
PAH	PROP	0.05	–	17.603	17.601 ± 0.009	1.0000	1.0000 ± 0.0000
PAH	PROP	0.1	–	21.496	21.500 ± 0.007	1.0000	1.0000 ± 0.0000
PAH	EXP	0.05	–	1.523	1.524 ± 0.004	1.0000	1.0000 ± 0.0000
PAH	EXP	0.1	–	3.252	3.253 ± 0.005	1.0000	1.0000 ± 0.0000

TABLE XI: Theory and Monte Carlo (MC) simulation of the relative extra revenue and the block redundancy ratio at the optimal attack parameters for $\alpha = 0.05$, with $W = 2$ for PPLNS and $\lambda_d = 2\lambda_1$ for EXP. MC results are the mean and the 99% confidence half-width over 20 independent runs of 10^7 attack cycles (T-PAW) or rounds (PAH).

Attack	Scheme	β	γ	r_1 (r_h) Theory	r_1 (r_h) MC	r_s Theory	r_s MC	r_u Theory	r_u MC	r_f Theory	r_f MC
T-PAW	PPLNS	0.05	0	0.0323	0.0323 ± 0.0000	0.0795	0.0795 ± 0.0001	0.0488	0.0486 ± 0.0002	0.0480	0.0478 ± 0.0003
T-PAW	PPLNS	0.05	0.5	0.0615	0.0615 ± 0.0000	0.1403	0.1404 ± 0.0002	0.0819	0.0818 ± 0.0001	0.0803	0.0801 ± 0.0003
T-PAW	PPLNS	0.1	0	0.0334	0.0334 ± 0.0000	0.0821	0.0821 ± 0.0001	0.0499	0.0499 ± 0.0001	0.0482	0.0482 ± 0.0002
T-PAW	PPLNS	0.1	0.5	0.0591	0.0591 ± 0.0000	0.1365	0.1364 ± 0.0001	0.0785	0.0785 ± 0.0001	0.0755	0.0755 ± 0.0002
T-PAW	PROP	0.05	0	0.0607	0.0607 ± 0.0000	0.1288	0.1289 ± 0.0004	0.0977	0.0975 ± 0.0008	0.5000	0.5000 ± 0.0000
T-PAW	PROP	0.05	0.5	0.0989	0.0989 ± 0.0000	0.2144	0.2148 ± 0.0006	0.1552	0.1554 ± 0.0003	0.5000	0.5000 ± 0.0000
T-PAW	PROP	0.1	0	0.0525	0.0524 ± 0.0000	0.1109	0.1110 ± 0.0002	0.0849	0.0849 ± 0.0004	0.3333	0.3333 ± 0.0000
T-PAW	PROP	0.1	0.5	0.0786	0.0786 ± 0.0000	0.1735	0.1737 ± 0.0003	0.1260	0.1260 ± 0.0001	0.3333	0.3333 ± 0.0000
T-PAW	EXP	0.05	0	0.1750	0.1750 ± 0.0000	0.3798	0.3798 ± 0.0000	0.2943	0.2945 ± 0.0006	0.5000	0.5000 ± 0.0000
T-PAW	EXP	0.05	0.5	0.2087	0.2087 ± 0.0000	0.4271	0.4271 ± 0.0000	0.3404	0.3402 ± 0.0003	0.5000	0.5000 ± 0.0000
T-PAW	EXP	0.1	0	0.1135	0.1135 ± 0.0000	0.2553	0.2553 ± 0.0000	0.1951	0.1951 ± 0.0003	0.3333	0.3333 ± 0.0000
T-PAW	EXP	0.1	0.5	0.1351	0.1351 ± 0.0000	0.2877	0.2877 ± 0.0000	0.2260	0.2260 ± 0.0002	0.3333	0.3333 ± 0.0000
PAH	PROP	0.05	–	0.3630	0.3630 ± 0.0000	–	–	–	–	–	–
PAH	PROP	0.1	–	0.2298	0.2298 ± 0.0000	–	–	–	–	–	–
PAH	EXP	0.05	–	0.0808	0.0808 ± 0.0000	–	–	–	–	–	–
PAH	EXP	0.1	–	0.0678	0.0678 ± 0.0000	–	–	–	–	–	–

TABLE XII: Theory and Monte Carlo (MC) simulation of the reward fractions at the optimal attack parameters, with the same setting as Table XI. For PAH, the first column gives the expected adversarial fraction r_h of a pool block. A dash marks an event that does not occur at the optimum.

APPENDIX F
MONTE CARLO VERIFICATIONS

To verify the analytical expressions, we simulate the attacks directly, without using any of the derivations in Appendices A–E. For T-PAW, we simulate many independent chains of attack cycles in parallel. Each chain follows the cases of Fig. 1 and keeps the state of its pool explicitly: the last W blockworks of pool shares, as a list of segments, for PPLNS; the blockwork of the open round for PROP; and the score of the open round, updated by Lemma 6, for EXP. Every canonical pool block pays the adversary according to the current state. For PROP and

EXP, we discard a burn-in of $\lceil 5/\beta \rceil + 10$ cycles so that the carry is stationary, and for PPLNS, we discard the cycles until the initial window is replaced. For PAH, the rounds are i.i.d. by Lemma 8, hence we simulate them independently. Since the numbers of outside and solo blocks in a round do not affect the following rounds, we replace them with their conditional means given the round length, which keeps the estimator unbiased and reduces its variance. For each case, we use the optimal parameters of Section VIII for $\alpha = 0.05$.

Table XI shows that the simulated RER and δ agree with the theory. For PAH, the confidence intervals are below 0.01%, and the theory and the simulation agree to the third digit, e.g., 17.603% versus $17.601\% \pm 0.009\%$ under PROP for $\beta = 0.05$. For T-PAW, the confidence intervals are wider, since the RER is a small difference between two revenue ratios, $\rho_A/\alpha - 1$, and the reward of a cycle has a large variance. The theory lies within the 99% confidence interval in all cases except PPLNS with $(\beta, \gamma) = (0.1, 0)$, where it lies just outside. Both this deviation as well as the fact that the T-PAW gain is of the same order as the confidence interval for PPLNS and PROP, imply these RER values alone are not a sharp test.

A sharper test is given by the reward fractions in Table XII. The revenue ratios of T-PAW depend on the payout scheme only through r_1, r_s, r_u, r_f, hence the scheme-dependent part of the analysis can be verified through these fractions alone. They are averages over many pool blocks, thus their confidence intervals are much smaller. The theory and the simulation agree to the fourth digit for every scheme, including the effect of the carry under PROP and EXP, e.g., $r_u = 0.1552$ versus 0.1554 ± 0.0003 under PROP for $(\beta, \gamma) = (0.05, 0.5)$. Under PPLNS, the simulated r_u and r_f are slightly below the theory, by at most 2×10^{-4}. At the optimum, $p_3 = 1$, hence the pool block of the fork race is paid $r_f = a'_3$ exactly under PROP and EXP, since it closes a fresh round. Finally, the simulated r_h of PAH matches the theory in every case, which verifies Theorem 6 and the blockwork age argument of Lemma 8.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, March 2008.

[2] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," 2011. [Online]. Available: https://arxiv.org/abs/1112.4980

[3] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, p. 95–102, July 2018.

[4] C. Grunspan and R. Pérez-Marco, "On profitability of selfish mining," 2019. [Online]. Available: https://arxiv.org/abs/1805.08281

[5] N. T. Courtois and L. Bahack, "On subversive miner strategies and block withholding attack in bitcoin digital currency," 2014. [Online]. Available: https://arxiv.org/abs/1402.1718

[6] L. Luu, R. Saha, I. Parameshwaran, P. Saxena, and A. Hobor, "On power splitting games in distributed computation: The case of bitcoin pooled mining," in *IEEE CSF*, July 2015.

[7] R. Qin, Y. Yuan, and F.-Y. Wang, "Optimal block withholding strategies for blockchain mining pools," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 3, pp. 709–717, 2020.

[8] wizkid057, "Block withholding attack against the eligius mining pool," https://bitcointalk.org/index.php?topic=441465.msg7282674#msg7282674, 2014, bitcointalk forum post reporting an estimated 300 BTC loss due to a block withholding attack.

[9] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin," in *ACM SIGSAC CCS*, October 2017.

[10] S. Gao, Z. Li, Z. Peng, and B. Xiao, "Power adjusting and bribery racing: Novel mining attacks in the bitcoin system," in *ACM SIGSAC CCS*, November 2019.

[11] M. Doger and S. Ulukus, "Incentive attacks in btc: Short-term revenue changes and long-term efficiencies," *IEEE Transactions on Network Science and Engineering*, vol. 13, pp. 11 434–11 459, 2026.

[12] D. Lee and W. Lee, "Return to dilemma: A novel block withholding attack on pow-based blockchains," *IEEE Transactions on Computational Social Systems*, vol. 12, no. 6, pp. 4141–4152, 2025.

[13] L. Liu, W. Chen, L. Zhang, J. Liu, and J. Qin, "A type of block withholding delay attack and the countermeasure based on type-2 fuzzy inference," *Mathematical Biosciences and Engineering*, vol. 17, no. 1, pp. 309–327, 2020.

[14] S.-Y. Chang, Y. Park, S. Wuthier, and C.-W. Chen, "Uncle-block attack: Blockchain mining threat beyond block withholding for rational and uncooperative miners," in *ACNS*. Springer-Verlag, June 2019, p. 241–258.

[15] J. Wang and Z. Wang, "Efaw: a new mining attack model combining faw attacks with the eclipse attack," *Journal of Surveillance, Security and Safety*, vol. 4, no. 4, pp. 180–195, 2023.

[16] Z. Yang, C. Yin, J. Ke, T. T. A. Dinh, and J. Zhou, "If you can't beat them, pay them: Bitcoin protection racket is profitable," in *ACM ACSAC*, December 2022, p. 727–741.

[17] J. Hu and N. Ruan, "Bm-paw: A profitable mining attack in the pow-based blockchain system," in *Blockchain and Trustworthy Systems*, 2026, pp. 3–17.

[18] H. Zhu, X. Chang, J. Mišić, V. B. Mišić, and R. Yang, "Revisiting faw attack in an imperfect pow blockchain system," *Peer-to-Peer Networking and Applications*, vol. 15, no. 5, pp. 2430–2443, Sep 2022.

[19] S.-Y. Chang, "Share withholding attack in blockchain mining: Technical report," 2020. [Online]. Available: https://arxiv.org/abs/2008.13317

[20] C. Grunspan and R. Pérez-Marco, "Block withholding resilience," *Digital Finance*, vol. 7, no. 1, pp. 43–60, 2025.

[21] Raulo, "Optimal pool abuse strategy," 2011, [Online]. Available: http://bitcoin.atspace.com/poolcheating.pdf.

[22] M. Belotti, S. Kirati, and S. Secci, "Bitcoin pool-hopping detection," in *IEEE RTSI*, 2018. [Online]. Available: https://cedric.cnam.fr/~seccis/papers/BeKiSe-RTSI18.pdf

[23] E. Cortesi, F. Bruschi, S. Secci, and S. Taktak, "A new approach for bitcoin pool-hopping detection," *Computer Networks*, vol. 205, p. 108758, 2022.

[24] N. Tovanich, N. Soulié, N. Heulot, and P. Isenberg, "An empirical analysis of pool hopping behavior in the bitcoin blockchain," in *IEEE ICBC*, 2021, pp. 1–9.

[25] O. Schrijvers, J. Bonneau, D. Boneh, and T. Roughgarden, "Incentive compatibility of bitcoin mining pool reward functions," in *Springer FC*, 2016, pp. 477–498.

[26] Y. Zolotavkin, J. García, and C. Rudolph, "Incentive compatibility of pay per last n shares in bitcoin mining pools," in *Decision and Game Theory for Security*, 2017, pp. 21–39.

[27] P. Lazos, F. Marmolejo-Cossío, X. Zhou, and J. Katz, "Rpplns: Pay-per-last-n-shares with a randomised twist," in *AAMAS*, 2021, p. 1569–1571.

[28] J. Chen and L.-Y. Wu, "Optimizing reward mechanism in pay-per-last-n-shares mining pools under block withholding attacks," *Blockchain: Research and Applications*, p. 100540, 2026.

[29] S. Zhu, W. Li, H. Li, C. Hu, and Z. Cai, "A survey: Reward distribution mechanisms and withholding attacks in bitcoin pool mining," *Mathematical Foundations of Computing*, vol. 1, no. 4, pp. 393–414, 2018.

[30] Y. Velner, J. Teutsch, and L. Luu, "Smart contracts make bitcoin mining pools vulnerable," in *Springer FC*, 2017, pp. 298–316.

[31] C. Grunspan and R. Pérez-Marco, "Profit lag and alternate network mining," in *Springer MARBLE*, 2023, pp. 115–132.

[32] mempool.space, "Bitcoin mining pools," https://mempool.space/mining, 2026, current Bitcoin mining pool hashrate distribution. Accessed: July 8, 2026.

[33] Foundry USA, "Foundry usa pool," https://pool.foundrydigital.com/, 2026, official mining pool website. Accessed: July 8, 2026.

[34] ANTPOOL, "Antpool," https://www.antpool.com/, 2026, official mining pool website. Accessed: July 8, 2026.

[35] ViaBTC, "Mining fees and payment methods," https://www.viabtc.com/en/pricing, 2026, official payout methods and fees. Accessed: July 8, 2026.

[36] f2pool, "Payout schemes: Pps, pplns, fpps, pps+," https://f2pool.zendesk.com/hc/en-us/articles/360061042332-Payout-schemes-PPS-PPLNS-FPPS-PPS, 2025, official documentation of supported payout schemes. Accessed: July 8, 2026.

[37] P2Pool, "P2Pool: Decentralized Monero mining pool, pool statistics," https://web.archive.org/web/20261008010810/https://p2pool.io/, 2026, archived snapshot, October 8, 2026.

[38] S. Chernykh, "P2Pool v3.0 release notes: Dynamic PPLNS window," GitHub release, https://github.com/SChernykh/p2pool/releases/tag/v3.0, 2023, hardfork of March 18, 2023: PPLNS window targeted to 2 Monero blocks (2 payouts) per share on average.

[39] "Comparison of mining pools," BitcoinWiki, revision 11312, section "Reward types", entry "Score". Accessed: 2026-10-20. [Online]. Available: https://bitcoinwiki.org/index.php?title=Comparison_of_mining_pools&oldid=11312

[40] K. A. Negy, P. R. Rizun, and E. G. Sirer, "Selfish mining re-examined," in *Springer FC*, 2020, pp. 61–78.

[41] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," in *Springer FC*, 2017.

[42] S. Asmussen, *Applied Probability and Queues*, 2nd ed., ser. Stochastic Modelling and Applied Probability. New York, NY: Springer, 2003, vol. 51.