Short Paper: PoSH
Proof of Staked Hardware Consensus

Rami Khalil⋆ and Naranker Dulay

Imperial College London
{rami.khalil, n.dulay}@imperial.ac.uk

Abstract. This paper introduces the PoSH Consensus protocol, a novel work-in-progress construction for achieving Sybil-resistant Nakamoto-style probabilistic consensus on the contents of a cryptocurrency ledger in a permissionless decentralized network where parties stake their hardware's computational power towards participation in leader election. PoSH aims to establish an openly mintable cryptocurrency that eliminates the requirement for block rewards and disincentivizes mining pools.

Keywords: Blockchain · Cryptocurrency · PoSH · Consensus.

1 Introduction

Decentralized cryptocurrencies have multiplied since the inception of Bitcoin [12]. However, most of them retain the core of Bitcoin's minting policy, only creating new coins in the form of block rewards to compensate block proposers. Such a policy maintains a recurring lottery that forms a cornerstone for Bitcoin's stability [4], where only the winner both receives the right to propose the next block, and claims the entire set of cryptocurrency units created in that block.

Several different technologies enable such lotteries fairly, allowing long term winning rates proportional to investment in participation. For example, Proof of Work (**PoW**) systems grant consensus lottery representation in proportion to computational hashing capabilities, but have generally been criticized for their large electricity requirements. Proof of Stake (**PoS**) systems grant consensus lottery representation in proportion to the percentage of ownership of a cryptocurrency without excessive electricity, but are prone to "long range" attacks. While one avenue of mitigating the effects of such attacks is to operate a Byzantine Fault Tolerant (**BFT**) consensus protocol atop PoS, another is to use Verifiable Delay Functions (**VDF**) [3] as Proofs of Sequential Work (**PoSeq**) [13,10] to delay block creation. Proof of Space (**PoSpace**) systems instead leverage PoSeq to grant consensus lottery representation in proportion to computational storage capacities while demanding much less electricity to function than PoW, yet suffer from lower resilience to adversarial presence in the network [5]. More comprehensive and detailed descriptions can be found in [1].

⋆ Supported by the Imperial College London President's PhD Scholarship

State-of-the-art systems offer a variety of very robust means of achieving consensus while employing block rewards and fixed currency minting rates. However, lottery-based block reward schemes do not quickly compensate their participants in proportion to their costs of participation, but require faithful ongoing participation for unknown periods of time before yielding appropriate rewards on average. This is especially problematic for participants with very small representation that may almost never win the lottery once in their entire lifetimes. Consequently, individual miners can band together to form a ”mining pool”, which allows them to combine their consensus participation powers and split the received rewards for more stable short-term gains. Furthermore, participants with a significant share of mining power may partake in ”selfish mining” attacks to take advantage of the variance in block reward distribution and maximize their gains. While some prominent works [14,2] tackle the latter two issues, no known mechanism promptly compensates small individual participants fairly.

On another front, recent proposals utilize PoSeq to operate alternative currency minting policies. Elasticoin [6] proposes publishing challenges on existing decentralized ledgers and computing proofs of sequential work on those challenges in a timely manner to mint new currency units. However, this design is restricted by the assumption that the underlying ledger will be able to confirm all proofs of sequential work on time, which creates a bottleneck. Melmint [7] employs the Elasticoin minting model to create a cryptocurrency pegged to the value of ”one day of sequential computation on an up-to-date processor”, with a circulating supply that responds to changes in demand. Similarly, a time-sensitive auction required to stabilize the currency’s value depends on the performance of the underlying ledger, which may present a bottleneck.

Proof of Staked Hardware[1] (PoSH) consensus aims to overcome the previously mentioned challenges through three design objectives:

1. Issue currency units in exchange for producing a PoSeq.
2. Enable consensus through staking newly minted currency units.
3. Reward block proposers only with transaction fees.

PoSH is essentially a PoW/PoS/PoSeq hybrid protocol with many unique features. Notably, PoSH makes no attempt to reduce energy consumption, and instead aims to optimize energy utilization, such that the majority of energy usage goes towards currency minting, while a relatively negligible amount goes towards consensus. PoSH’s design choices result in many interesting system characteristics, as discussed throughout the paper.

This short paper proceeds as follows: Section 2 reviews the cryptographic primitives used. Section 3 presents an overview of the core details of PoSH consensus, describing the minting, staking, mining and block proposal processes. Section 4 briefly discusses the potential advantages of PoSH. Section 5 overviews potential future work, while Section 6 concludes this short paper.

[1] The name is derived from the fact that the computations of hardware are staked towards participation in consensus. The risk in staking is clarified in Section 3.

2 Preliminaries

Verifiable Delay Functions. The following is a summary of the VDF formalization as presented first in [3]. In short, a VDF is comprised of three algorithms:

1. $\mathrm{Setup}(\lambda, t) \rightarrow pp$, which takes a security parameter λ and a delay parameter t and outputs the public parameters pp required to evaluate and verify the VDF such that, for all inputs x, it is hard to find a y for which the outputs of the other two algorithms $\mathrm{Verify}(pp, x, y, \pi) = \top$, and $\mathrm{Eval}(pp, x) \neq y$.
2. $\mathrm{Eval}(pp, x) \rightarrow (y, \pi)$, which takes the public parameters pp and a value x from the input domain and outputs a value y in the output range, and (optionally) a proof π. Honest parties must be able to compute y in t sequential steps, while no adversary with a polynomial number of parallel processors can distinguish y from a random output range sample in significantly fewer steps.
3. $\mathrm{Verify}(pp, x, y, \pi) \rightarrow \{\perp, \top\}$, which validates in $O(\mathrm{polylog}(t))$ steps whether y is the output for x under pp given π.

Throughout the rest of this work, it is assumed 1) that $\mathrm{Setup}(\lambda, t)$ is executed for some known values of λ and t, and 2) that pp is published. Furthermore, $\mathrm{Eval}(pp, x)$ is simply referred to as $f^t(x)$. Remarkably, any system which can prove the evaluation of t unique computations, not necessarily sequential, generated from a known input x can be used to instantiate PoSH, while in this paper, PoSH is simply explained in terms of VDFs for concreteness.

Misc. A secure cryptographic signature scheme, and a collision-resistant cryptographic hashing function H are assumed to be known.

3 Protocol Overview

From a bird's-eye view, a party that wants to participate in PoSH consensus first has to mint new currency units; then stake these units by simply not spending them; subsequently use its stake to run a local throttled mining procedure; and finally propose a block if it manages to win the consensus lottery.

Minting. A VDF input and evaluation output pair $(x, f^t(x))$ is referred to as a thread. A single PoSH unit of currency is valued at one work step. The list of valid inputs x for minting is unique to every party in the system. It is generated from hashing the minter's public identity, the hash of a valid block, along with either a nonce or the output of its parent thread if one exists. This domain restriction ensures that no party can derive $(f(x), f^{t+1}(x))$ from $(x, f^t(x))$ then present it as a separate thread worth t work steps, and that each thread is bound to the identity of its creator, a block in the chain, and optionally a parent thread.

From the perspective of a minter, the value of a thread is equal to its production cost, which is a combination of time, electricity, and hardware costs. The parameter t essentially dictates the value of a single thread in terms of currency units, and remains fixed. It must be noted that different processors will be able to complete t steps in different time-spans that possibly differ by orders of magnitude. Therefore, t should be set such that even a theoretically highly optimized piece of hardware should take a non-negligible amount of time to evaluate $f^t(x)$. The VDF Alliance[2] is a notable coalition seeking to build open source hardware that computes VDFs at high throughputs, and such highly optimized hardware could ultimately serve as a practical reference point for a PoSH instance.

Staking. The staking mechanism in PoSH is designed to leverage freshly produced threads as a requirement for participation in consensus.

A thread is considered *fresh* if it is either not more than E_B blocks old, or the timestamp of the block used as its input for minting is not more than E_T seconds old. More importantly, this requires that the block whose hash was used as input for creating the thread is part of the chain the thread is being staked on. The freshness property dictates that all stake is temporary, which entails that a party must continuously create new threads to maintain participation power, effectively putting a continuous demand on its hardware resources. This leads to two interesting features:

1. Unlike PoS protocols, and more like PoW protocols, there is no notion of ”initial stake distribution” in PoSH. Stake can be created permissionlessly at will by any capable party.
2. ”Stake hoarding” becomes infeasible, as a party’s expected fresh stake value after a large amount of time can be modelled as the expected length of a queue where the arrival rate is that of the party’s thread production, and the departure rate is that of the party’s thread expiry.

A thread is staked by using it as input to activate a local instance of the throttled mining procedure described later. Interestingly, parties need not declare their intention to stake their freshly minted units in any way, and only run the mining procedure locally. Moreover, a single party with n threads staked has the same number of mining instances running as n parties with a single thread staked each. These two properties could potentially protect the privacy of a stakeholder, who can opt to switch identities with each thread.

The two expiry parameters, E_B and E_T, are closely tied to the parameters t, and another parameter V_r, which is referred to in this paper as the *reference forward velocity*, measured in terms of work steps per second towards evaluation of $f^t(x)$. For the purposes of staking, $E_T = \frac{t}{V_r}$. Given $E[\Delta_T]$ as the targeted average inter-arrival time of blocks, $E_B = \frac{E_T}{[\Delta_T]}$.

Mining. PoW mining can be used to enable PoSH mining, but must operate at a forcibly throttled rate as to drastically reduce wasted work. First, F_s is defined as the scaling factor, which is how many work steps are required per PoW mining attempt. Combined with V_r, the hashing rate of a throttled mining instance is defined as $H_r = \frac{V_r}{F_s}$ hashes per second.

[2] `https://www.vdfalliance.org/`

Given a relatively fresh complete thread $(x_j, f^t(x_j))$, the throttled mining procedure attempts a deterministic sequence of nonces based on $f^t(x_j)$ to mine the block B_i. The k^{th} nonce may be attempted depending on the time since the last block. Given V_r, and F_s, the k^{th} nonce may be attempted if $0 < k < \Delta_T \times H_r$, where Δ_T is the difference between the party's current time and the timestamp of block B_{i-1}. If the the k^{th} nonce is acceptable, the timestamp T_i of the newly mined block B_i must be set to the timestamp of block B_{i-1} plus $\frac{k}{H_r}$ seconds. This gives the added benefit of mitigating potential block timestamp forgery, and ensuring that whatever difficulty adjustment algorithm is employed receives accurate information as a result of the throttled mining procedure.

An acceptable nonce is a classic PoW, but does not target the current block being mined. Instead, when hashed with block number j, an acceptable nonce produces a value that is below the mining target threshold. j is the latest block number where any thread instantiated as of block j, or earlier, is ineligible to mine on block i (not relatively fresh as of block i). This lag leverages the dependence of fresh coin validity on block hash validity to restrict parallel chain growth, and prevent "stake grinding", as an adversary that attempts to check how a block it proposes will affect its future leader election chances must first exhaust its computational resources to mint new threads on top of that block.

Difficulty adjustment also plays a major role in the mining process, but does not require a tailored algorithm. Notably, mining difficulty corresponds with the number of relatively fresh threads actively used in mining, which corresponds with the network minting rate.

Block Proposal. Once a viable nonce is found, its discoverer may sign a new block B_i, and publish it to the network. However, the network must not accept or propagate blocks with timestamps too far into the future, depending on H_r.

The only rewards for the block proposer are the collected transaction fees. This is because the overwhelming majority of the cost borne to participate in consensus should be that of minting, rather than mining or staking.

If a party signs, and propagates, multiple blocks at the same height, their identity is simply blacklisted only for the current height, and all their parallel blocks are dropped only if they form the tip of the longest chain. Otherwise, there is no penalty for attempting to fork the chain.

Despite this lax policy on how to respond to attempted forking, mining on the longest chain should be the only sound option for an honest miner, mostly due to the fact that an active participant needs to choose new blocks to mint on top of as soon as they finish the thread they are currently creating. If a miner deviates from mining on top of the longest chain, it risks losing its ongoing participation power in the network, as it will not be able to decisively choose the next block for thread creation. This can be seen as a delayed version of the mining power allocation decision in PoW.

However, a miner that does not intend to renew its stake, and is effectively exiting the network, effectively has nothing at stake when mining on multiple chains. How long such a miner poses a threat to the network largely depends on E_B and E_T. Consequently, this aspect deserves a much more rigorous analysis.

Transactions Transactions can spend previously existing balances as in any other blockchain using a signature from the account owner, but a transaction issued by a minter which redeems or spends a newly minted thread must be accompanied by the proof π that t work steps were performed by the minter to create that thread. Efficient batch verification of thread evaluation would lower the cost of spending newly minted units. A batch of threads is simply referred to as a batch from now on.

In case a batch verification mechanism is absent or is inefficient, a probabilistic sampling technique can be used instead. First, the outputs of all threads in a batch are committed to using a vector commitment[3]. Then, the commitment is used to derive a fixed-size pseudo-random sequence of threads from the batch for verification, given a security level and a discount factor. For example, given a security level of 2^{128}, a batch can be spent for 95% of its value after verifying that 1730 random thread samples were correctly computed. The remaining 5% are discounted, or burned, and cannot be trusted to have been performed due to the constraints of the statistical method employed. While the security level must remain the same for all minters, each may choose their own discount factor in order to maximize the utility of a batch based on the cost of verifying the required number of samples. For example, a batch may be processed to yield only 80% of its value, but incur the cost of verifying only 398 samples if verifying an additional 1332 samples costs more than 15% of its value.

The structure of a batch plays an important role in the economics of transaction spending. A batch is defined by its length and width, which denote the length of each chain of threads in the batch and the number of such chains, respectively. Notably, to keep minters focused on the latest blocks, a batch may not be spent unless it is relatively fresh, which requires that the last threads of all chains in a batch be relatively fresh.

4 Discussion

Inflation. The inter-block currency inflation rate r_i at B_i, given that the total issued supply as of B_{i-1} equals s_{i-1} and the units issued as of B_i equals m_i, is defined as $r_i = \frac{m_i}{s_{i-1}}$. This means that over time, given a steady minting rate m per block, the inflation rate after n future blocks will equal $r_{i+n} = \frac{m}{s_{i-1}+n \times m}$. Simply put, currency supply inflation approaches minting rate inflation over time. Interestingly, the global minting rate, excluding minters not participating in mining, can be estimated from the mining difficulty, V_r and $E[\Delta_T]$.

Privacy. Threads that do not result in an acceptable PoW when staked can be kept hidden until they are spent. This introduces uncertainty about total currency supply and individual account balances. Moreover, this enables a form of private payment where a payer can simply mint units on a recipient's thread. Introducing zero-knowledge transactions, as in Zcash [9], could enable even further privacy, and possibly reduce the cost of thread batch verification.

[3] such as a simple Merkle tree.

Smart-contracts. The possibility to mint PoSH units that can be directly credited to a smart-contract, in a party's name, could have groundbreaking potential for applications and scalability, particularly for off-chain, or second-layer, payment protocols, surveyed in [8], as it would mean that one could deposit funds into a smart-contract by directly minting units for it rather than performing a transaction in its favor using the ledger.

Inclusion. It is extremely unlikely to win a block reward while independently mining many cryptocurrencies at home using one's personal computer as of today. While minting PoSH units will take variable times on different processors, it remains a guaranteed task that can be accomplished after a known number of work steps, regardless of network difficulty. However, while the cost effectiveness of spending newly minted threads depends on network fees, the thread count required to cost-effectively spend a batch can always be estimated.

Valuation. Roughly speaking, aside from stable coins [11], almost every cryptocurrency today suffers severe volatility in pricing due to relying on scarcity and speculation for value. PoSH units, on the other hand, can be very conveniently subjectively valued by an individual, since one can always choose to mint, rather than buy, if market prices exceed expected minting costs, while selling below production cost entails incurring a known loss.

5 Future work

Formalization. The description of PoSH provided therein can best be described as an informal overview of the core novelty of the protocol. For implementation, analysis and extension purposes, a more comprehensive specification should be given. Fortunately, such a specification is not far from being available.

Implementation. Luckily, various implementations of the cryptographic building blocks of PoSH are readily available in the public domain. Therefore, a working implementation of PoSH, an endeavor already in pursuit, would constitute a valuable contribution, and enable further analysis of the protocol.

Analysis. A formal security analysis of the composition of PoSH is needed to rigorously define its guarantees. Moreover, a game theoretic analysis is needed to investigate whether honest protocol behavior is indeed rational, and a quantitative analysis is needed to determine the effects of the different parameters of PoSH and reach a method to potentially optimize them.

Extension. All five components presented in Section 3 are not set in stone. An instantiation of PoSH with other minting, staking, mining, block proposal, or transaction methods could yield a protocol with alternative desirable properties.

6 Conclusion

Prior to this work, the concept of a consensus protocol completely based on an openly, yet privately, mintable currency without any separate governance token had been unexplored. This paper lays the core building blocks of such a construction, and outlines the future work required to rigorously understand PoSH, and possibly similar hybrid protocols. The PoSH principle of optimizing energy utilization through currency minting, while minimizing consensus effort, is a promising technique that presents a plethora of opportunities for innovation. Furthermore, as PoSH seems to exhibit the characteristics of both PoW and PoS protocols, it may be a step towards a generalization that captures both families.

References

1. Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., Danezis, G.: Sok: Consensus in the age of blockchains. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies. pp. 183–198 (2019)
2. Bissias, G., Levine, B.N.: Bobtail: Improved blockchain security with low-variance mining. In: ISOC Network and Distributed System Security Symposium (2020)
3. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Annual international cryptology conference. pp. 757–788. Springer (2018)
4. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 154–167 (2016)
5. Dembo, A., Kannan, S., Tas, E.N., Tse, D., Viswanath, P., Wang, X., Zeitouni, O.: Everything is a race and nakamoto always wins. arXiv preprint arXiv:2005.10484 (2020)
6. Dong, Y., Boutaba, R.: Elasticoin: Low-volatility cryptocurrency with proofs of sequential work. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). pp. 205–209. IEEE (2019)
7. Dong, Y., Boutaba, R.: Melmint: trustless stable cryptocurrency. Cryptoeconomic Systems (2020)
8. Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: Sok: Off the chain transactions. IACR Cryptol. ePrint Arch. **2019**, 360 (2019)
9. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification. GitHub: San Francisco, CA, USA (2016)
10. Long, J., Wei, R.: Nakamoto consensus with verifiable delay puzzle. arXiv preprint arXiv:1908.06394 (2019)
11. Moin, A., Sekniqi, K., Sirer, E.G.: Sok: A classification framework for stablecoin designs. In: Financial Cryptography (2020)
12. Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system.(2008) (2008)
13. Orlicki, J.I.: Sequential proof-of-work for fair staking and distributed randomness beacons. arXiv preprint arXiv:2008.10189 (2020)
14. Pass, R., Shi, E.: Fruitchains: A fair blockchain. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 315–324 (2017)