Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake

Iddo Bentov*
Computer Science Dept., Technion
idddo@cs.technion.ac.il

Charles Lee
Litecoin Project
coblee@litecoin.org

Alex Mizrahi
chromawallet.com
alex.mizrahi@gmail.com

Meni Rosenfeld
Israeli Bitcoin Association
meni@bitcoin.org.il

Abstract

We propose a new protocol for a cryptocurrency, that builds upon the Bitcoin protocol by combining its *Proof of Work* component with a *Proof of Stake* type of system. Our *Proof of Activity* (PoA) protocol offers good security against possibly practical future attacks on Bitcoin, and has a relatively low penalty in terms of network communication and storage space. We explore various attack scenarios and suggest remedies to potential vulnerabilities of the PoA protocol, as well as evaluate the performance of its core subroutine.

1 Introduction

The Bitcoin [33] cryptocurrency continues to gather success since its launch in 2009. As a means of exchange, Bitcoin facilitates fast worldwide transactions with trivial fees and without identity theft risks. As a store of value, Bitcoin entails no counterparty risk and no exposure to manipulation of the money supply by central banks, due to its decentralized nature. Many other features can be derived from Bitcoin's cryptographic foundations, including: resilient forms of secret sharing via multi-signature control over an address [13, 18], non-interactive transactions with other systems via computational integrity proofs [29], off-chain micropayment channels [20, 36], various types of contracts that are enforced by the distributed network [18], trust-free gambling and multiparty computation [1, 2, 4], trust-free sale of computed data via zero-knowledge proofs [24], hierarchical wallets that use key homomorphism to minimize access to the private keys [26], decentralized stock exchange and prediction market via colored coins [12, 32], and so on.

Thus it is natural to ask which kinds of attacks on the Bitcoin network are likely to be practical, and which ideas would be the most effective in mitigating plausible attacks on a successful cryptocurrency. Since Bitcoin has proven to be quite capable of resisting attacks up until now, our focus is on long term sustainability. More specifically, we are especially concerned with the attack environment after *Proof of Work* (PoW) mining is no longer subsidized via the block reward, and the network needs to be secured via transaction fees acquired from the commerce taking place.

A robust cryptocurrency protocol should strive to provide an incentives structure under which it is in the self-interest of the different participants in the system to sustain its health. In this work, we offer an elaborate extension to the Bitcoin protocol as a remedy to what we argue to be probable security threats, in the sense that an attack on the extended protocol would be much more expensive. Essentially, our analysis and proposed remedy stem from the proposition that PoW miners do not have the right economic incentives to be solely in charge of securing the network, and that stakeholders are fitted to assist in this task. The *Proof of Activity* (PoA) protocol that we propose is also likely to have other features, such as promoting an enhanced network topology and less overall energy consumption. In exchange for the desirable properties

*Supported by funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 240258.

that we argue that the PoA protocol achieves, nodes in the PoA network are required to do more work and communication relative to nodes in the Bitcoin network.

The purpose of the PoA protocol is to have a decentralized cyrptocurrency network whose security is based on a combination of *Proof of Work* and *Proof of Stake*. In general terms, *Proof of Work* based protocols give the decision-making power to entities who perform computational tasks, while *Proof of Stake* based protocols give the decision-making power to entities who hold stake in the system. While we contend that *Proof of Stake* based protocols offer consequential advantages, *Proof of Stake* is neither trouble-free nor effective at mitigating all the major risks that a successful cryptocurrency faces. One major risk is centralization, as data centers that are dedicated to PoW computations and transactions verification may outcompete hobbyist miners, due to economies of scale. With *Proof of Stake* systems, the particular risk of PoW data centers is indeed reduced, though other risks remain intact (see Section 2.1), and new kinds of centralization risks are introduced (large stakeholders may try to exhibit control over the system, as well as the more subtle risks that are presented in Section 5). Another major risk is an erosion of fungibility due to blacklisting of "tainted" coins.[1] This risk is orthogonal to *Proof of Stake*, and could be mitigated by mixing [11, 28] or SNARKs [14, 30].[2]

Different *Proof of Stake* systems have been envisioned since 2011, in the context of an improvement to Bitcoin. The first discussion was probably [42], and concrete protocols were initially considered at [40]. We contend that the earlier proposals have security vulnerabilities or impractical overhead, or milder benefits compared to the PoA protocol. See Section 3.1 and [5] for more details.

Let us give here a succinct description of the Bitcoin protocol (see [6, 33] for additional details). We use a similar language here and in Section 3, to allow for easy comparison between Bitcoin and PoA.

1. Each miner collects transactions that are broadcasted over the network, and uses her hashpower to try to generate a block via repeated invocation of a hash function on data that consists of the transactions that she saw fit to include, the hash of the previous block, her public key address, and a nonce.
2. When a miner succeeds in generating a block, meaning that the hash of her block data is smaller than the current difficulty target, she broadcasts her block to the network.
3. In case other miners see that this block is valid, i.e. it references the hash of the previous block and meets the current difficulty target, and see that it is the longest[3] extension of the chain of blocks (a.k.a. blockchain) that they are aware of, they move on to continue to extend the blockchain from this block.

- The block reward (newly minted coins) and the fees from the transactions that the miner collected go to the public address that she provided. This means that only the miner can spend the coins that she earned, by signing with her corresponding private key.
- The difficulty level readjusts according to the mining power that participates, by updating the hash target value every 2016 blocks ($\approx$ 2 weeks) so that blocks get generated once every 10 minutes on average.
- The block reward starts at 50 coins and halves every 210,000 blocks, i.e. about every 4 years.

Figure 1: Outline of the Bitcoin block creation process

The organization of this paper is as follows. In Section 2 we describe attacks on Bitcoin that are purportedly feasible. In Section 3 we specify our extension of the Bitcoin protocol and discuss its characteristics. In Section 4 we consider another important aspect of a cryptocurrency, namely fair distribution of the initial money supply, in the context of our PoA protocol. In Section 5 we compare between attacks on PoA and on Bitcoin, and analyse potential vulnerabilities of the PoA scheme. In Section 6 we estimate the cost difference

[1] See for example http://www.forbes.com/sites/kashmirhill/2013/11/13/sanitizing-bitcoin-coin-validation/

[2] Current SNARK-based designs are incompatible with *Proof of Stake* protocols such as PoA.

[3] Longest not in total number of blocks, but measured in work difficulty without distinction between blocks that belong to the same retarget window of 2016 blocks.

between attacks on PoA given certain parameters, vis-a-vis an attack on Bitcoin. In Appendix A we run preliminary benchmarks to measure the performance of the critical subroutine of PoA, and discuss future efficiency enhancements.

2 Motivation

There are different ways in which direct attacks on Bitcoin's pure PoW protocol can be attempted. One kind of an attack is the infamous >50% hashpower attack, where the attacker invests in hardware equipment (ASIC) to obtain more PoW hashpower than all the other Bitcoin miners combined [6, 33, 34]. Such an attacker could then double-spend by reversing the recent ledger history to defraud merchants, or carry out a PoW-denial-of-service (PoW-DoS) attack by refusing to include transactions in the blocks that she generates, unless perhaps the transactions conform with the policy that the attacker imposes. In case the attacker is malicious and wishes to destroy or harm the Bitcoin network, she may achieve her objective because either double-spending or PoW-DoS can cause a loss of confidence in the Bitcoin protocol. In the case of a greedy self-interested attacker, she can make direct financial gains via double-spending, or extort others and demand higher transaction fees by carrying out PoW-DoS.

If an attacker has enough resources to obtain >50% of the total hashpower, then it is not unreasonable to assume that she could also obtain e.g. 90% of the total hashpower, which would increase the effectiveness of the PoW-DoS attack. While it is true that the attacker depletes her resources as she carries out PoW-DoS, and therefore the Bitcoin network can survive this attack by simply waiting until the attacker gives up, in practice there could be a snowball effect where honest miners quit as confidence in the network is being lost, thus making it easier for the attacker to obtain the vast majority of the total hashpower.

Let us also note that an attacker who obtains a substantial minority of the total hashpower can also plausibly attempt double-spending attacks [33, 34].

2.1 The attack environment that arises from the Tragedy of the Commons

It is likely that PoW mining will become significantly less lucrative when the block reward subsidy becomes negligible, and the reward consists (almost) entirely of transaction fees. The rationale behind this stems from an economic phenomenon known as the "Tragedy of the Commons" [17], which is a prevalent problem in the study of economics and game theory. In its most general form, it is a system in which participants have an opportunity to act selfishly, taking action to benefit themselves at the cost of harming their peers. A selfish rational agent will always take such action because she is interested only in her own well-being; but if everyone acts selfishly, everyone will be worse off than if everyone cooperates. Any agreement to mutually cooperated for the benefit of all will not be stable, because every rational participant will prefer to enjoy the fruit of both everyone else's cooperation and her own defection.

There are several instantiations of this phenomenon in relation to funding the operation of the Bitcoin network. It is in the interest of every Bitcoin user that fees will be paid for transactions, to encourage miners to provide the network with a sufficient level of security; however, each user will prefer that others pay fees, while he pays no fee and still enjoys the network security. Without a way to force users to pay fees, the vast majority of the users will avoid paying, and end up with an insecure system that is of use to no one.

The solution lies within the power miners have to reject transactions if the fee paid is not high enough. However, here we have another tragedy of the commons problem, between the different miners. Without protocol-enforced limitations on what can go into a block, a rational miner will prefer to include every fee-paying transaction, even if the fee is very low, because the marginal cost of including a transaction is trivial. If miners accept low-fee transactions, users will have no reason to pay significant fees, and the total fees that can be collected by miners will not be sufficient to cover the cost of PoW mining.

The miners could try to form an agreement to accept only high-fee transactions; in this scenario, users will be forced to pay high fees if they want their transaction included (meaning, fees that are high enough for an adequate network security, while low enough for the market to bear - which is desirable). However, the agreement is not stable - it is in the interest of each miner to defect and accept low-fee transactions,

as this action boosts the revenue of this individual miner. If all the miners do so, the bargaining power of miners will be eroded and they will no longer be able to force high fees, reducing the total revenue.

Since the total mining revenue is what funds the overall network security, in this case the network security will be weak. For this reason, maintaining a healthy network requires some protocol-enforced rules protecting miners, as a group, from themselves - such as, a cap on the total value transferred in transactions in each block. If the cap is properly chosen, miners will actually earn more with this kind of a cap in place - by making block space a scarce resource, its price goes up; transactions will have to compete with others for admission, and pay high fees for the privilege. An individual miner cannot break the market by accepting low-fee transactions, as she can only put so many in the block.

For example, let us say there exist in the market 1000 users wanting to send a transaction of 1 BTC each, and willing to pay 1% of the transaction value in fees (the amount per transaction is irrelevant as long as the total BTC and percentage fee stay the same). In addition, there are 1000 users wishing to send 1 BTC, for whom the transaction is not as important and so they are only willing to pay 0.1% in fees. If there is no cap, every miner will want to include all transactions, even if their fee is only 0.1%. There is no way to effectively segment the market, so with 0.1% fee transactions being accepted, this is what will be paid also by those willing to pay more. The total transaction value will be 2000 BTC and the fee paid is 0.1%, for a total of 2 BTC. If, however, a cap is placed at 999 BTC, the miners are no longer free to include all transactions. They must choose which 999 transactions to include, and the equilibrium is that those users who can bear it will pay 1% (if they pay less, a user whose transaction is excluded will offer a higher fee to get ahead of the others, and so on). Thus, 999 BTC will be transacted but with a fee of 1%, and the total revenue is 9.99 BTC.

Still, if the total cost of PoW mining at an adequate security level is more than what the market can bear, just having a protocol-enforced rule such as a block value cap will not be enough. When the transactions volume (and market cap) of a cryptocurrency increases, the total amount of transaction fees will increase too (as derived from the fees that the users can bear), and therefore it may seem that this total fees amount will be sufficient to fund the security of the network. However, as the market cap of the cryptocurrrency grows, the incentives to attack it also increase, hence the cost of maintaining the security of the network is correlated with an increased market cap and transactions volume. Thus, we need a way to have a high ratio of security to transaction fees. The growth of a *Proof of Work* based cryptocurrency should not be expected to make this ratio better, unlike *Proof of Stake* based protocols such as PoA, because the entities that secure the PoA network have fewer expenses and may therefore collect lower fees (due to competition among them). Stated differently, since the overall network security is proportional to the total transaction fees paid, users may disagree on which protocol rules offer the best tradeoff between security and low fees, and *Proof of Stake* relieves some of this tension as network security requires less computational effort and therefore has a lower cost attached to it.

There is also a third tragedy of the commons problem, as the transaction fees are paid only to the miner who created the block, while the cost of propagating, verifying, and storing the transactions is shared by all the nodes in the network. Miners will prefer keeping every transaction to themselves and collecting a fee for it, while avoiding as much as possible the work of propagating it. Having a value cap for each block will not help in this regard, because the users as a whole may still wish to send a very large amounts of low-value transactions. Here the solution is limits on data size and CPU cycles (currently dominated by ECDSA signature verifications) for each block, which is controversial since many users believe that the block size should accommodate the Bitcoin economy. *Proof of Stake* based protocols offer little help here, as they do not reduce these particular costs.

In the overall scheme of things, a negligible block reward subsidy is likely to imply an environment in which it is easier to carry out PoW-based attacks, because there would be fewer honest miners to compete against. In particular, for a governmental entity who wishes to destroy the competition against the fiat currency that it issues, a clandestine ASIC-based attack could be quite easy. An entity who carries out PoW-based attacks under these circumstances is not likely to possess any significant amount of bitcoins, since Bitcoin stakeholders would be scrambling to keep the network secure in order to protect their fortune. Moreover, the PoW hardware may retain some resale value to the attacker, and also to other miners who

prefer to exchange the bitcoins that they earn for fiat currencies and hence have no paramount interest in the soundness of the cryptocurrency. Note that ASIC mining hardware may have resale value too, in particular since it can be repurposed to mine alternative cryptocurrenies. Self-interested miners may even delegate their (ASIC) hashpower to a service that automatically switches among the most profitable cryptocurrencies to mine[4], implying that such miners are oblivious as to whether their hashpower participates in attacks.

Therefore, it makes sense to vest part of the power that synchronizes the transactions in the hands of the stakeholders, rather than vesting all of this power in the hands of the PoW miners.

Let us note that other kinds of remedies to this perceived danger have also been proposed, in particular network assurance contracts [19].

2.2 Additional types of hazards that a cryptocurrency faces

Another category of direct attacks is network attacks, in particular network denial-of-service and network isolation. Here the topology of the online nodes in the Bitcoin network is attacked. When the connectivity between the nodes is low, it becomes easier to deny service by flooding miner nodes, or carry out a Sybil attack by isolating and transacting with some specific node [25]. Due to pooled PoW mining[5], the current topology of the Bitcoin network is barely at the level of a single popular torrent [16]. In contrast, as we will see in Section 3, the PoA protocol incentivizes non-miner nodes to maintain a continual online presence, and this should be quite helpful to the overall network topology. Even if all the Bitcoin miners were using p2pool[6], the Bitcoin network topology would probably still be worse than the PoA network topology, because many Bitcoin users are not miners. Note that miners who have a relatively low hashrate may consider the variance of p2pool to be too high, and therefore opt to use more centralized schemes such as multi-PPS [37].

There are also indirect attacks that can be effective. For one, an attacker may try to corner the market on Bitcoin. This attacker could manipulate the price at the exchanges in which Bitcoin is traded, causing a loss of confidence. With *Proof of Stake* based protocols such as PoA, stakeholders are less likely to succumb to downward price spirals, because the coins that they hold generate revenue proportional to the actual commerce taking place. Just as an attempted hostile takeover to corner the market on a certain stock is good for its shareholders, an attempt to corner the market on PoA is good for its stakeholders (who, in analogy with shareholders, have voting power).

Of course, a cryptocurrency is also susceptible to various other kinds of governmental [23] and private [6, 21] attacks and hazards. Some of the plausible threats are not sensitive to the distinctions between *Proof of Work* and *Proof of Stake*, and are therefore beyond the scope of this discussion.

3 Protocol

The *Proof of Activity* protocol is an extension of the Bitcoin protocol. The network nodes in PoA need to do more complex verifications compared to the work that the Bitcoin network nodes do, and our contention is that this extra work brings about some particular benefits.

The primary subroutine that PoA incorporates is called *follow-the-satoshi*, whereby we transform some pseudorandom value into a satoshi (smallest unit of the cryptocurrency) that is picked uniformly among all the satoshis that have been minted thus far. This is done by selecting a pseudorandom index between zero and the total number of satoshis in existence up to the last block, inspecting the block in which this satoshi was minted, and following each transaction that transferred this satoshi to a subsequent address until reaching the address that currently controls this satoshi. Note that this process can be regarded as picking a pseudorandom stakeholder in a uniform fashion, e.g. if Alice has 2 coins and Bob has 6 coins then Alice is 3 times less likely to be picked compared to Bob.

In Figure 2 we specify how blocks are generated in the PoA network.

[4] See for example `http://middlecoin.com`, `https://hashco.ws`, etc.
[5] See the distribution among centralized PoW pools at `http://blockchain.info/pools`
[6] Decentralized pooled mining protocol, see `https://en.bitcoin.it/wiki/P2Pool`

1. Each miner uses her hashpower to try to generate an empty block header, i.e. header data that consists of the hash of the previous block, the miner's public address, height relative to the genesis block (a.k.a. the index of the block in the blockchain), and a nonce. This header does not reference any transactions.

2. When a miner succeeds in generating an empty block header, meaning that the hash of her block header data is smaller than the current difficulty target, she broadcasts her block header to the network.

3. All the network nodes regard the hash of this block header as data that deterministically derives N pseudorandom stakeholders. The derivation is done by concatenating this hash with the hash of the previous block and with N fixed suffix values, then hashing each combination, and then invoking *follow-the-satoshi* with each of the N hashes as input.

4. Every stakeholder who is online checks whether the empty block header that the miner broadcasted is valid, meaning that it contains the hash of the previous block and meets the current difficulty. Upon validation, the stakeholder checks whether she is one of the N lucky stakeholders of this block. When the first $N-1$ lucky stakeholders discover that the block derives them, they sign the hash of this empty block header with the private key that controls their derived satoshi, and broadcast their signature to the network. When the N^{th} stakeholder sees that the block derives her, she creates a wrapped block that extends the empty block header by including as many transactions as she wishes to include, the $N-1$ signatures of the other derived stakeholders, and her own signature for the hash of this entire block.

5. The N^{th} stakeholder broadcasts the wrapped block to the network, and when the other nodes see that this wrapped block is valid according to the above, they consider it a legitimate extension of the blockchain. The nodes try to extend the longest branch of the blockchain that they are aware of, where "longest" is measured in PoW difficulty as in Bitcoin.

- The fees from the transactions that the N^{th} stakeholder collected are shared between the miner and the N lucky stakeholders (see Section 4 for details).

Figure 2: The PoA block creation process (think $N = 3$)

3.1 Discussion of the protocol

Let us first note that if some of the N lucky stakeholders were offline, then other miners will also solve the block and thereby derive N other pseudorandom stakeholders, so the overall difficulty will readjust both according to the total hashpower and according to what fraction of all the stakeholders is online.

The block creation process of PoA is accomplished in two rounds of communication, unlike Bitcoin's single round of communication. In fact, if we set $N = 1$ instead of amplifying the stakeholders' power by picking $N > 1$ winners, PoA will also require just one round of communication. Either way, PoA's block creation process is significantly more involved than Bitcoin's block creation process.

Note that picking the lucky stakeholders via a scheme that measures the Hamming distance between some pseudorandom value and a valid address would be susceptible to PoW attempts at creating fresh addresses, or blockchain bloat as stakeholders would split their coins among a huge number of addresses.

The PoW component of the protocol is essential for throttling the derivation attempts at picking lucky stakeholders, as otherwise the derivations will occur at blazing speeds. One underlying problem is that without the PoW component, there is no clear way to make the system convergent, i.e. to have a protocol rule that specifies the weight of a valid branch in a way that both preserves decentralization and gets the network to converge on the winning branch, without introducing feasible attack vectors. Another problem is that rational stakeholders can maximize their expected reward by signing every forked branch that they see, if PoW is not required to extend branches. In a followup work [5] we contrast PoA with existing pure *Proof of Stake* systems, as well as with novel pure *Proof of Stake* protocols.

The protocol can allow the N pseudorandom stakeholders to move their coins to a new P2SH address [9] instead of sending only an auxiliary signature, to retain the 160-bit security against a second preimage attack,

compared to the 128-bit security that 256-bit ECDSA keys imply. The stakeholder's client may decide to attach the additional 160-bit address only if the amount of coins in the output is above some threshold. Thus each block will contain up to $\approx N \cdot 160$ extra bits.

Let us point out that there exists a similarity between picking a pseudorandom stakeholder uniformly, and giving voting power to groups of stakeholders. For example, if there are two groups of stakeholders, one honest group that controls 80% of the stake and one malicious group that controls 20% of the stake, then picking a pseudorandom stakeholder uniformly implies that the malicious group is picked 20% of the time and therefore it effectively has 20% of the voting power. In comparison to other *Proof of Stake* schemes where many stakeholders would use their voting power to attach their signatures to certain checkpoint blocks in order to solidify the blockchain, the PoA protocol has substantially less overhead in terms of network communication and blockchain bloat.

The word "activity" in the phrase *Proof of Activity* emphasizes the point that only active stakeholders who maintain a full online node get rewarded, in exchange for the vital services that they provide for the network. This stands in contrast to earlier *Proof of Stake* schemes in which offline stake can accumulate weight over time, and may ultimately be utilized in double-spending attacks.

We have assumed that the transactions are standard, i.e. that each output is redeemable by signing with one private key that corresponds to the one specified address, so it is straightforward for a stakeholder to provide an auxiliary signature that proves that she can spend the winning satoshi. The Bitcoin protocol includes a scripting language that allows forming more complicated nonstandard transactions that could be spent via multiple signatures [18], or without any signatures at all [24]. Hence there is a contradiction between *Proof of Stake* and nonstandard scripts, as a stakeholder who can redeem a nonstandard output might not be able to provide a signature that proves that she controls that output. Fortunately, many of the use cases of nonstandard transactions involve speedy trades rather than long term holding. We propose that nonstandard scripts must be of the rigid form (A_0 or $(\ldots)$) to be eligible for the stakeholders' lottery, i.e. redeemable either by signing with a private key sk_0 that corresponds to the address A_0, or according to the rest of the script. Nonstandard scripts that are compatible with this form can be useful, e.g. (A_1 or (A_2 and A_3)) where sk_1 is controlled by a secure hardware wallet, sk_2 resides on a smartphone, and sk_3 resides on a laptop. See Section 3.2.3 on why having in each output an auxiliary public key that may only be utilized upon winning the lottery is a bad idea.

Any particular choice of N for the number of derived stakeholders could be criticized as a "magic number", much like the 10 minute block time of the Bitcoin protocol. Opting for a greater number implies that the power of the stakeholders relative to the miners is more amplified, though it also implies an increased communication complexity and data bloat. The question of whether this number could be dynamic rather than constant is legitimate, although it is far from clear whether the dynamic option is advantageous overall. Having a dynamic number of derived stakeholders could be possible by using similar ideas to Section 3.2.1, i.e. the protocol could detect the stakeholders' participation level and specify that the number of derived stakeholders gets lowered when the participation level is too low, and vice versa. The drawbacks of the dynamic option are more complicated protocol rules, and more importantly the security threats that arise as an attacker might be able to exploit these protocol rules by having less need to use stake in the branch that she generates in secret. Similarly, replacing the constant 10 minute block time target with a dynamic block time could also expose the cyrptocurrency to possible vulnerabilities [22,35,43]. On balance, we regard the constant $N = 3$ for the number of derived stakeholders as a quite reasonable option. See Section 5 for details.

The PoA protocol rewards stakeholders who participate and sustain the network, rather than punish stakeholders who do not participate. Arguably, there are potential advantages to *Proof Of Stake* protocols that punish passive stakeholders, namely a greater encouragement for stakeholders to pitch in, and a fairer wealth distribution. This last concern alludes to "the rich get richer" effect, i.e. it would be easier for a stakeholder who holds a lot of coins to be rewarded with even more coins. Though taken to the extreme, the stakeholder's coins offer her no value unless she spends them, i.e. this phenomenon is comparable to an interest bearing bank account. The concern is not necessarily much different with PoA than with Bitcoin,

i.e. a PoW miner can use the coins that she earns to buy more mining hardware. The similarity exists because (as mentioned) the expected reward that results from *follow-the-satoshi* is linear in the stake held, and the expected reward from PoW mining is linear in the hardware costs, meaning that e.g. a miner who operates two identical mining machines is expected to earn twice as much as with one of the machines. That being said, the PoA protocol can be augmented with a punishment mechanism, by deriving additional pseudorandom stakeholders who should provide their signatures in one of the next several blocks, or else be punished via coin confiscation. However, one major pitfall with this kind of a non-deadlocking signature solicitation component is the perverse incentive to use centralized stake pools (see Section 5.2), therefore we do not incorporate such a mechanism into the PoA protocol.

As is the case with the Bitcoin network, nodes in the PoA network may have a disincentive to re-transmit data for the blocks that are being created. To maximize their expected reward, stakeholders may refrain from re-transmitting transactions (if $N = 1$), and miners and stakeholders may refrain from re-transmitting empty PoW block headers. When the number of online nodes is small as a result of pools, this concern is rather mild, since the users of the cryptocurrency can transmit data directly to the pools and hence guarantee speedy confirmations for their transactions. However, one of the objectives of PoA is to have many online stakeholders. When the number of online stakeholder nodes is large, each such node gets rewarded infrequently, and will therefore expect the marginal loss of reward by re-transmitting data to be small. Thus, data will still be re-transmitted by altruistic stakeholders who follow the protocol, and by some of the rational stakeholders according to how they weigh the small expected loss of reward against the value of their stake as dictated by the overall health of the network. Beyond that, the PoA protocol can be augmented with methods that encourage data propagation [3], by rewarding the nodes that re-transmit a transaction with a portion of the transaction fee.

The reason for concatenating the hash of the previous wrapped block when we derive the N lucky stakeholders, instead of using just the hash of the current empty block header, is that the entropy[7] of a block header hash decreases as the PoW difficulty level rises. But this is a minor issue, e.g. the hashPrevBlock field of Bitcoin's block header consists of multiple leading zeros, which is adequate since the remaining entropy is still significant enough. Moreover, the PoW difficulty level of PoA is expected to be lower than that of Bitcoin (see Section 6), so it can be deemed reasonable to eliminate this extra concatenation.

3.2 Add-ons to the protocol

Here we provide several add-ons to the basic PoA protocol, that are intended to improve the overall soundness of the cryptocurrency.

3.2.1 Incentivizing a target participation level

The N^{th} stakeholder who creates the wrapped block may include inside it not only the regular transactions, but also all the other empty blocks that the miners broadcasted, i.e. empty block headers that extend the previous block but do not derive her as the lucky stakeholder who is responsible for creating this block. Notice that a compact empty block for this purpose can be just the pair (public_address, nonce), and the network nodes can check its validity by verifying that hash(nonce, hash(previous_block_hash, public_address, height)) meets the required PoW difficulty target[8].

The benefit of this add-on is that the protocol can measure the stakeholders' participation level. At each difficulty retarget window, the nodes could now sum up the amount of empty blocks that were generated since the last difficulty retarget. This way the protocol can specify that the relative reward of the stakeholders should increase during the next retarget windows in case their participation level is too low, and vice versa.

For example, if 50% of the stakeholders are online and $N = 3$, we expect the stakeholder who creates the block to include 7 additional empty block headers inside it. Therefore, in case we see at the retarget window

[7] By entropy we mean the minimum average number of bits needed to represent an output.

[8] This hashing method is slightly more efficient and accommodates pooled mining, otherwise hash(previous_block_hash, public_address, height, nonce) suffices.

that on average more than 7 empty headers were added on to each block, we conclude that the stakeholders' participation level is less than 50%. Assuming that the PoA protocol specifies a 50% participation target[9], the protocol rule would dictate here that a higher ratio of the fees goes to the stakeholders relative to the miners during the next retarget window. This should incentivize the network to always converge to at least 50% stakeholders' participation level.

To encourage stakeholders to include the extra empty blocks, the stakeholder who creates the block may receive small fees for adding these empty block headers, in the form of a higher ratio of the total reward. Though the higher reward ratio that would benefit all the stakeholders during the next retarget window may be enough of an incentive.

3.2.2 Discouraging thin clients by *Proof of UTXO*

One way to combat greedy or malicious miners is to force them to prove that they are capable of validating transactions via *Proof of UTXO* [31], i.e. by extending the PoW hash function to be a memory-hard hash function that derives pseudorandom queries to the current set of unspent coins (UTXO set) from the input seed. In PoA we gain a similar benefit, because participants in the stakeholders' lottery who run full nodes need to access the UTXO set and update it according to the transactions in the latest blocks. Depending on the optimizations used (see Appendix A), the stakeholder's node may also be an archival node , i.e. a node that can bootstrap new nodes by serving them the entire history in a trust-free manner.

Greedy stakeholders may opt to use a "thin" client that does not maintain the UTXO set, and instead expect the miner who solved the empty block to invoke *follow-the-satoshi*, collect valid transactions, and broadcast a message suited for thin clients that lists the N derived stakeholders and contains the wrapped block that needs to be signed. It may not be straightforward for an attacker to exploit such thin clients, as they may utilize competing services that strip the empty header that miners broadcast and construct a different wrapped block from it.

The PoA protocol can force stakeholders to maintain the UTXO set, by deriving queries to the current UTXO set from a signature of the block, and requiring the resulting hash to meet a certain PoW difficulty. Thus, if the N^{th} stakeholder does not have the UTXO set, she will need to spend an impractical amount of communication to create the wrapped block. This form of *Proof of UTXO* is quite energy efficient, as the required PoW difficulty can be very mild.

3.2.3 Limited-withdrawal key delegation

One disincentive for a stakeholder to operate an online node is that her client software has to sign the block with the private key that controls the satoshi that won the lottery, which means that if her computer is compromised then the attacker can gain access to her private keys. By contrast, PoW miners are not exposed to this risk. The simple way to eliminate the risk is by extending the protocol to let stakeholders delegate their signing power to special keys that are allowed to sign upon winning the stakeholders' lottery, but not allowed to spend coins by signing regular transactions. This way, the private keys can be kept in encrypted form, as the online nodes that participate in the lottery would not need to utilize them. However, the unintended consequence of such a protocol extension is that stakeholders would be more likely to give their delegated signing keys to centralized entities, as described in sections 5.2 and 5.3.

We propose a middle-of-the-road approach, whereby the delegated key can periodically be used to transfer a limited but substantial amount of coins to any address, and can always be used to move all the coins to a single designated address. Stakeholders who wins the lottery can use this limited-withdrawal private key to sign the newly created block, which reduces the risk that the online nodes are exposed to.

Let R and T be two constants. We add to the protocol a *delegation transaction* with which a stakeholder can move coins from her address A_0 to the special address (A_1, A_2), i.e. we assume that this stakeholder has the privates keys sk_0, sk_1, sk_2 that correspond to the addresses A_0, A_1, A_2. The limited-withdrawal key

[9] Participation targets that are close to 100% could be possible, but then the protocol should provide for a method to revive lost coins and thus achieve zero monetary deflation.

sk_1 can be used to sign when (A_1, A_2) wins the stakeholders' lottery, to move all the coins to A_2, and to create a transaction that spends up to R fraction of the coins to arbitrary outputs and the rest of the coins back to (A_1, A_2). Every transaction that transfers coins from (A_1, A_2) to arbitrary addresses has an implicit `lock_time` [10] of $h + T$ blocks, where h is the height of the block that contained the (A_1, A_2) input. This entails a waiting period of T blocks before being able to spend another R fraction of the coins that (A_1, A_2) still controls.

With 10 minute blocks, we consider $R = 10\%$ and $T = 144$ to be reasonable choices. For example, suppose that the stakeholder holds 500 coins. An attacker who obtained the delegated private key sk_1 can spend 50 coins, and another 45 coins after about one day, and so on. This should discourage the stakeholder from giving sk_1 to a centralized entity. The stakeholder can subscribe to a notification service that will alert her as soon as the first 50 coins were stolen, so that she could immediately use sk_1 to move the remaining 450 coins to A_2.

Let us remark that the limited-withdrawal feature is useful in other contexts besides the stakeholders' lottery. In fact, under ordinary use of sending payments to merchants, private keys can be stolen by keyloggers etc., hence keeping the full fledged private keys offline and encrypted while using just the limited-withdrawal keys can be a good practice.

3.2.4 Decentralized stake pools

The protocol can support a kind of a contract between participants of a stake pool, that splits the reward among them without involving trust.

Let us depict the idea by example. Suppose that Alice holds 4 coins, Bob holds 3 coins, and Carol holds 2 coins. They sign a flagged transaction that transfers their 9 coins to a keyless master address, and broadcast this transaction to the network. For the purpose of *follow-the-satoshi*, if this master address wins the lottery with a satoshi that arrived through Alice's address, then the network allows only Alice to provide the signature, and the protocol rule is that the profits that are earned by creating the block are given directly to that addresses of Alice and Bob and Carol in a way that rewards Alice more than the others, e.g. Alice gets twice as much as Bob and Carol. This bigger reward to Alice is done in order to discourage her from doing a withholding attack on the pool, i.e. to stay offline and reap the rewards that Bob and Carol bring. If Alice wishes to leave the pool, she can sign a transaction (which requires a fee) that transfers her 4 coins back to herself, and the rest of the 5 coins to another keyless master address that specifies Bob and Carol as its owners.

It could still be the case that only one of the pool participants runs a full online node, by communicating with the other participants who maintain an online presence in order to provide their signatures when needed. This would be worse for the pool in terms of network lag when there are tight races to win blocks. In any case, there is no designated pool operator.

Due to the overhead, this kind of decentralized stake pools may not be suitable for small stakeholders. These stakeholders may extend their trust to a centralized stake pool (c.f. Section 5.2), where the pool operator participates in a larger decentralized pool, and accounts for the coins that she receives from the small stakeholders as she decides how much of her reserves should be put in the decentralized pool.

4 Money supply

With pure *Proof of Stake* cryptocurrencies, distributing the coins to the interested parties in fair manner is less straightforward than with PoW cryptocurrencies. For example, it is informative to observe the hardships that Ripple[10] runs into as it handles the initial distribution of its built-in coin [38].

With PoA, we have the benefit of the PoW aspect that is incorporated into the system, which can be used for handling the initial distribution of the coins. However, if the PoA protocol specifies that the block reward subsidy is divided about equally between the miner and the N lucky stakeholders, starting from the

[10] Decentralized payment system that is based on social networks, see `https://ripple.com`

genesis block, then this is likely to enable the rich to get richer in an unfair manner. One alternative is to use a pure PoW protocol until the first block reward halving after 4 years, and only then roll out the full PoA scheme. Another alternative is to always give the entire reward subsidy to the PoW miner who solved the block, and share the transaction fees between this miner and the N lucky stakeholders. This may imply that users will have to pay nontrivial transaction fees starting from the genesis block, in order to incentivize stakeholders to run full online nodes. However, it is reasonable to expect that the fees paid to stakeholders would not be excessive. This should mean that the added incentive to hoard will be small, i.e. the fees can be a nice added bonus if the stakeholder wishes to save the coins anyway, but if she has alternative uses for the wealth then these fees will not be enough to make her hold.

The apportionment can be specified according to certain constants. The portion that goes to the N^{th} stakeholder should be relatively big, unless perhaps if all the N lucky stakeholders must maintain the UTXO set (see Section 3.2.2). E.g., with $N = 3$ the protocol can dictate that $\frac{1}{2}$ of the reward goes the miner, $\frac{1}{4}$ goes to the 3^{rd} stakeholder, and $\frac{1}{8}$ goes to each of the two other stakeholders. The apportionment can also be dynamic, in accordance with Section 3.2.1.

5 Security

The PoA protocol vests some of the power that generates the blocks in the hands of the stakeholders. As we will demonstrate, an attacker needs to control very large amounts of stake in addition to PoW hashpower if she wishes to double-spend by preparing her branch in secret before broadcasting it to the network.

Furthermore, by letting a lucky stakeholder determine which transactions should be included in the block, PoA provides security against attackers who would try to extort or destroy the network by denying transactions. Other forms of protection from this attack appear to be quite murky. For example, there can be a protocol rule that requires at least M amount of BDD [7, 27] in each block, but this would imply: (1) M has to be a significant amount, otherwise the attacker could create blocks with transactions in which she always just transfers small amounts of coins among her own addresses, therefore (2) if it is a slow day and you wish to send some small amount of coins, you might have to wait for a long time until M BDD is reached, in other words we would lose the predictability of 10 minute blocks, and more importantly (3) if the cryptocurrency's economy grows to e.g. $1000M$ BDD in each block on average, then the attacker could carry out her attack by creating blocks with only M BDD in them. Alternatively, the protocol rule can weigh more favorably branches with more BDD, meaning that a branch with less PoW height but more BDD weight than another branch may still win. However, this introduces the attack vector in which an attacker waits patiently until she accumulates a large amount of BDD, and then exploits this rule to double-spend by outcompeting the average BDD of the honest network.

Let us demonstrate how the use of N lottery winners in the protocol amplifies the power of the honest stakeholders, which in turn diminishes the effectiveness of PoW-dominated attacks.

Assumption 1. The function that takes a block and derives from it values that are used as inputs for *follow-the-satoshi* is a random oracle.

Claim 1. If Assumption 1 holds, an attacker with x fraction of the *online* stake needs to have more than $(\frac{1}{x} - 1)^N$ times the hashpower of the honest miners in order to gain an advantage over the network.

Proof. Let $E_1 = \{$the N lucky stakeholders that the block derives are under the attacker's control$\}$ and $E_2 = \{$the N lucky stakeholders that the block derives are honest$\}$. We condition on the event $E_3 = \{$the N lucky stakeholders that the block derives are online$\}$, and note that $\Pr[E_1|E_3] = x^N$ is the probability that N online stakeholders that a mined block derives are under the attacker's control, and that $\Pr[E_2|E_3] = (1-x)^N$ is the probability that N online stakeholders that a mined block derives are honest. This means that on average the attacker will generate a block after $(\frac{1}{x})^N$ nonce attempts that meet the current difficulty target and derive N online stakeholders, while the honest network needs $(\frac{1}{1-x})^N$ such attempts on average. Therefore, if the attacker is fast enough so that she could compute $(\frac{1}{x})^N/(\frac{1}{1-x})^N = (\frac{1}{x} - 1)^N$ nonce attempts

per one nonce attempt of the honest network, she can generate the blocks at the same average speed as the rest of the network. □

Claim 2. If Assumption 1 holds and p fraction of the honest stake is online, an attacker with y fraction of the *total* stake needs more than $((\frac{1}{y}-1)\cdot p)^N$ times the hashpower of the honest miners in order to gain advantage over the network.

Proof. Similarly to Claim 1, this follows because the speedup factor that the attacker needs is $((1-y)\cdot p)^N/y^N = ((\frac{1}{y}-1)\cdot p)^N$, where $((1-y)\cdot p)^N$ is the probability that N derived stakeholders are both online and honest, and y^N is the probability that N derived stakeholders are controlled by the attacker. □

Remark. We disregard the small potential advantage that a dedicated attacker may have over the honest network in case she has local access to all of her stake (and PoW hashpower), which implies that she avoids the need to broadcast empty block headers and thus experience network lag. For example, if we assume a rather pessimistic average propagation time of 15 seconds [15], with 10 minute blocks we would adjust the speedup factor that the attacker needs by $\frac{600-15-15}{600} = 0.95$. The first 15 seconds compensate for propagating the empty block header, assuming that the miners keep trying to re-solve the block and never sit idle, so only one solved header should be factored in. The other 15 seconds compensate for broadcasting the finalized wrapped block, as the network nodes still work on the previous block until the new block propagates to them. In a pure PoW network, we need to compensate by a factor of $\frac{600-15}{600} = 0.975$, so the attacker gains an advantage with >49.36% instead of >50% of the hashpower under these assumptions. If the attacker does not personally own all of the stake that is used in the attack, but instead colludes with remote stakeholders by stealth (see Section 5.3), this advantage becomes even more negligible.

For example, if $N = 3$ and we assume that the transaction fees and limited-withdrawal key delegation incentivize $p = 50\%$ participation level of the honest stakeholders, an attacker who has 88.8% of the total hashpower and $y = 20\%$ of the total stake would still not have an advantage over the honest network: $((1/\frac{20}{100}-1)\cdot\frac{50}{100})^3 = 2^3$, meaning that the attacker needs to be more than 8 times faster than the rest of the network. If we adjust by the aforementioned 0.95 factor, the attacker will need 88.37% instead of 88.88% of the total hashpower.

Let us clarify why an attacker who tries to deny transactions (as much as she can) does not gain an additional advantage by creating empty blocks also when the hashpower and the first $N-1$ derived stakeholders are not controlled by her. If a large fraction of the blocks still gets generated by other stakeholders, she simply forgoes her rewards and the network continues to function, albeit more slowly. Hence, for an effective attack, she should seek to create consecutive empty blocks. However, if she creates empty blocks in the honest branch as the N^{th} winner with the help of any $N-1$ winners, she indeed weakens the usefulness of this branch, but at the same time she strengthens this branch in the race against the branch that consists of consecutive empty blocks that she is building.

We note that the larger N is, the possibility for a bribe attack becomes increasingly tangible. The reason is that the attacker can operate a scheme that bribes stakeholders to withhold their signatures on the honest chain, and promises to pay only the first derived stakeholder who withheld, per block. The attacker can try to reduce her costs by differentiating between offline stakeholders and online stakeholders who participate in the attack, though this may require communication between the attacker and the bribe-takers, which entails the risk that the double-spending attack becomes public.

With a PoW-based cryptocurrency, the security is sustained under the assumption that the majority of the mining power that participates is honest. Similarly, the PoA network derives its soundness from the assumption that the majority of the online stake is honest. Due to the amplification via the parameter N, the security of PoA deteriorates quickly when the majority of the online stake is under the control of a malicious entity. In Section 2.1 we argue that for a cryptocurrecy to be attack-resistant over the long term, relying on the assumption that the majority of the stake is honest is more conservative than to rely on the assumption that the majority of the hashpower is honest.

In Section 6 we summarize some comparative figures to illustrate the cost of an attack on PoA, subject to the stakeholders' participation level p and the parameter N. We compare it with the estimated cost of an attack on Bitcoin, and consider how much energy usage is required to keep PoA and Bitcoin secure.

5.1 Network denial of service

As with Bitcoin, the PoA network is resistant to DoS attackers who broadcast many invalid blocks, since it is easy to verify whether the wrapped block contains the public key preimages of the addresses of the N lucky stakeholders. Nodes may consider P2SH [9] preimages that are too large to be ineligible.

In case the N^{th} lucky stakeholder is an attacker who attempts to broadcast many invalid blocks with N valid signatures, the nodes can easily resist the DoS attack by blacklisting the empty block header that derived this N^{th} stakeholder.

5.2 Centralized stake pools

One potential problem is that similarly to PoW pooled mining, stakeholders will trust their coins in the hands of a centralized entity. A stakeholder might wish to do that both in order to decrease the expected time and variance until she collects her reward, and because she does not wish to be bothered by the effort and be exposed to the risks that are associated with running a full node. One common case is online wallet sites, as inexperienced users may reasonably assume that an online wallet can secure a small amount of coins better than they can. In exchange for this kind of service, the stakeholder will let the centralized stake pool keep some of the reward as a fee.

The risk that a stakeholder takes by entrusting her coins with the centralized stake pool is significantly greater than the risk that a miner takes when she delegates her PoW hashpower to a mining pool, because if e.g. the miner already holds 1000 coins in total and she is expected to earn 2 coins within a week by delegating her hashpower to the pool, she risks losing just the 2 coins, compared to the stakeholder who risks losing all of the coins that she holds. The risk of losing all the coins is an inherent characteristic of the stakeholders' lottery: we cannot have a centralized stake pool that operates in a similar manner to centralized PoW pools by letting its members provide the signatures by themselves when they win the lottery, and awarding shares to the pool members by quizzing them occasionally with random data to sign in order to verify that they are online, because the stakeholder who wins the lottery can keep the reward instead of sharing it with the pool. The underlying reason here is that the pool's reward address (c.f. coinbase [8]) cannot be set in advance (compare with Section 3.2.4).

The crucial source of concern is that both the miner and the stakeholder run the equally important risk that the centralized entity will abuse the power that was delegated to it and attack the network, which is plausible if the centralized entity becomes too big.

Let us illustrate the payout delay with concrete numbers. Suppose that a stakeholder holds 5000 coins (or perhaps the 5000 coins are held by a small group of stakeholders who trust each other and decide to cooperate), and the total amount of coins held by online stakeholders is 10 million. With 10 minute blocks, this stakeholder will collect her reward after less than two weeks on average, which should be good enough to discourage her from risking an entire wealth of 5000 coins with the centralized stake pool, as well as paying it a fee. As long as each such centralized pool controls e.g. $\frac{5000}{10000000} < 0.1\%$ of the online stake, the overall p2p network should still be quite decentralized and secure.

5.3 Double-spending bribes service

Another possible concern is a centralized entity that provides a double-spending service. Here the entity runs a stealth operation, where it recruits stakeholders who would collude with it by agreeing to provide their signatures to blocks in a secret branch that would subsequently be broadcasted, in order to double-spend. The revenues of this operation will come from bribe payments by its customers, who wish to reverse their blockchain transactions. The service would share the profits that were obtained via the double-spending attacks with the stakeholders who colluded with it.

The distinction between recruiting stakeholders for this job and recruiting PoW miners for this job is that the miners waste their resources in case the double-spending attack failed, while the work that the stakeholders do is almost costless. However, the collusion between this centralized entity and the stakeholders that it recruits should be secretive. If the entity openly advertises its server to solicit many stakeholders, then the merchant could also connect and listen on this server, and refuse to send her goods if she sees that a hostile branch is being formed. In case the entity does not wish to reveal the block data (height in particular) in its competing branch, it will need to waste more rounds of communication by first asking the derived stakeholders to sign a fixed message, then send the entire header only to the $N-1$ stakeholders who responded, then receive their signatures and prepare the wrapped block, and then send the entire wrapped block to the N^{th} stakeholder and wait for her signature. Note that the entity cannot send just a hash for the stakeholders to sign, since they should refuse as otherwise they can be tricked into signing a transaction that spends their coins. Further, if the server is public then it can be DoS-attacked by responding to its signature solicitations with many junk signatures. Hence, for example with $N = 3$, the bribes service needs to collude in secret with stakeholders who hold 10% of the total stake and control 98.9% of the hashpower, to become faster than a network in which 50% of the stakeholders participate (see Section 6).

Our discussion of a single attacker rather than multiple attackers is without loss of generality. For example, consider a joint effort by two separate double-spending services, each controlling 20% of the total stake. The two services must collude with each other in a secretive manner while preparing the hidden branch, or else the merchant could detect their activity. So these two stealth entities can be regarded as a single attacker with 40% of the total stake.

Suppose that instead of running a clandestine server, the attacker starts e.g. 6 blocks behind when it overtly attempts to solicit stakeholders. Let x the fraction of the online stake that the attacker controls, y the fraction that is self-interested, z the fraction that is honest i.e. follows the PoA protocol, and let w be the attacker's fraction of the total hashpower. The double-spending service should then be able to sustain >50% of the block creation power over a potentially long period of time, i.e. these unlikely conditions can be sufficient for the attack to succeed:

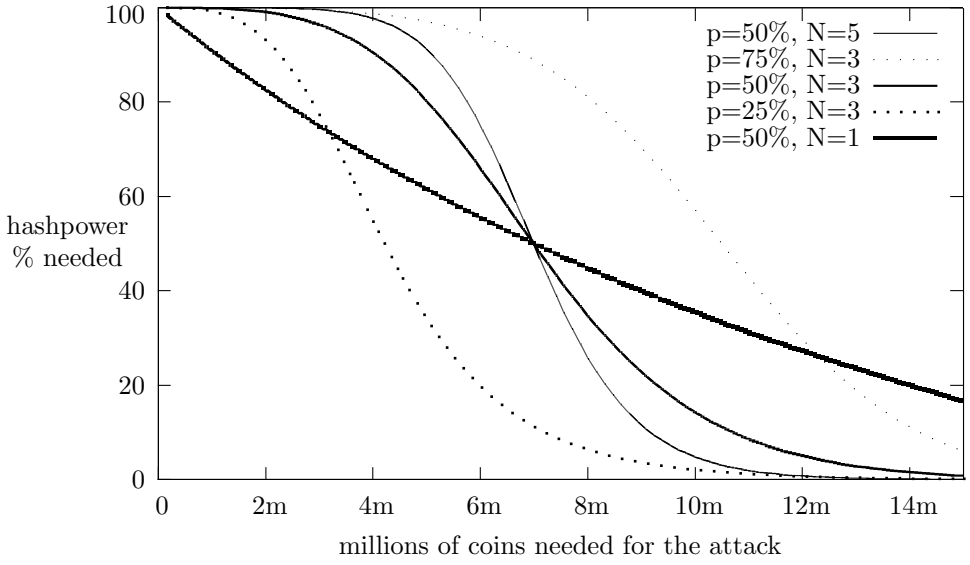
1. All of y wishes to also sign the attacker's branch.
2. $\frac{w}{1-w} > (\frac{z}{x})^N$, for example $w > 50\%$ and $x \geq z$ (c.f. sections 5 and 6).

Note that condition (1) is unlikely because stakeholders do not wish to have their stake diminish in value due to double-spending attacks. The attacker may thus try to bribe stakeholders, which makes the attack more costly.

6 Cost analysis

Let us compare the costs of an attack on PoA by giving exemplary figures in the table and chart below. The attacker's expenses depend on the honest stakeholders' participation level p and the amplification parameter N. We assume that the coins needed for the attack are out of a total of 21 million minted coins.

N	attacker's % of online stake	attacker's % of total stake	stakeholders' participation	coins needed	speedup needed	hashpower % needed
3	10%	5.2%	50%	1.1m	729	99.8%
3	18.1%	10%	50%	2.1m	91.1	98.9%
3	33.3%	20%	50%	4.2m	8	88.8%
3	40%	25%	50%	5.2m	3.3	77.1%
any	50%	33.3%	50%	7m	1	50%
3	25%	20%	75%	4.2m	27	96.4%
1	10%	5.2%	50%	1.1m	9	90%
1	18.1%	10%	50%	2.1m	4.5	81.8%
1	33.3%	20%	50%	4.2m	2	66.6%
5	33.3%	20%	50%	4.2m	32	96.9%
2	33.3%	20%	50%	4.2m	4	80%
2	40%	25%	50%	5.2m	2.2	69.2%
3	9.1%	1%	10%	210k	970.2	99.8%
1	9.1%	1%	10%	210k	9.9	90.8%
3	52.6%	10%	10%	2.1m	0.72	42.1%
1	52.6%	10%	10%	2.1m	0.9	47.3%
3	71.4%	20%	10%	4.2m	0.06	6%



To get a rough idea of the cost difference between an attack on Bitcoin and an attack on PoA, let us take for example an `AntMiner S2` ASIC unit that runs at 1 terahash/s and costs about 8 coins. Currently the total hashrate of the Bitcoin network is around $50,000$ terahash/s, therefore an attacker needs to have under her control $\approx 50,000$ `AntMiner` units that cost $400,000$ coins, in order to have $\frac{1}{2}$ of the total hashrate. Contrast that to e.g. 4.2 million coins that an attacker needs to control in order to have 20% of a total stake of 21 million coins, for gaining just $\frac{1}{3}$ of the online stake in a network in which 50% of the honest stakeholders participate. If we take $N = 3$ and assume that the hashrate of the PoA network is for example $\frac{1}{10}$ of Bitcoin's pure PoW network, i.e. around 5000 terahash/s, then this attacker also needs to control about $40,000$ `AntMiner` units with a price tag of $320,000$ coins, in order to be 8 times faster than the honest miners in the PoA network. Keep in mind that if the total hashrate of the PoA network is indeed $\frac{1}{10}$ of Bitcoin's pure PoW network, then PoA is much more efficient in terms of energy consumption.

Our focus was on the cost of gaining an advantage over the honest network. See the tables in [33, 34] regarding the attacker's prospects when she is at a disadvantage relative to the rest of the network. E.g., with $N = 3$ and $p = 50\%$, an attacker who has 20% of the total stake and 50% of the total hashpower is 8 times slower than the honest network, therefore her odds of carrying out a successful double-spending attack that reverses the last 6 blocks are about 0.1%.

7 Conclusion

The PoA protocol seeks to decentralize the power that synchronizes the transactions in a quite pronounced fashion. To monopolize the block creation process, an attacker needs to control a substantial fraction of the total amount of coins that have been generated thus far. We argue that in likely scenarios the cost of an attack would be much higher with the PoA protocol than with Bitcoin's pure PoW protocol. Furthermore, the PoA protocol is likely to accomplish other beneficial properties, namely an improved network topology, incentives for maintaining full online nodes, low transaction fees, and a more efficient energy usage.

Acknowledgments. We thank "cunicula" (pseudonym) for many useful discussions regarding *Proof Of Stake* in general, and PoA in particular. We thank Gregory Maxwell for numerous useful insights on Bitcoin, as well as cryptocurrencies more broadly. We thank Lear Bahack for critique regarding several aspects of the security analysis of PoA.

References

[1] Andrychowicz M., Dziembowski, S., Malinowski D., and Mazurek, L. 2014. Secure multiparty computations on bitcoin. In *IEEE Symposium on Security and Privacy (S&P)*.

[2] Andrychowicz M., Dziembowski, S., Malinowski D., and Mazurek, L. 2014. Fair two-party computations via the bitcoin deposits. In *First Workshop on Bitcoin Research, Financial Cryptography*.

[3] Babaioff, M., Dobzinski, S., Oren, S., and Zohar, A. 2012. On bitcoin and red balloons. In *ACM Conference on Electronic Commerce*, B. Faltings, K. Leyton-Brown, and P. Ipeirotis, Eds. ACM, 56–73.

[4] Bentov, I. and Kumaresan, R. 2014. How to Use Bitcoin to Design Fair Protocols. In *Advances in Cryptology — Crypto 2014*. http://eprint.iacr.org/2014/129.

[5] Bentov, I., Gabizon, A., and Mizrahi, A. 2014. Cryptocurrencies without Proof of Work. *Preprint*. http://www.cs.technion.ac.il/~idddo/CoA.pdf.

[6] Barber, S., Boyen, X., Shi, E., and Uzun, E. 2012. Bitter to better - how to make bitcoin a better currency. In *Financial Cryptography*, A. D. Keromytis, Ed. Lecture Notes in Computer Science Series, vol. 7397. Springer, 399–414.

[7] Bitcoin wiki. Bitcoin days destroyed. https://en.bitcoin.it/wiki/Bitcoin_Days_Destroyed.

[8] Bitcoin wiki. Coinbase. https://en.bitcoin.it/wiki/Coinbase.

[9] Bitcoin wiki. P2SH. https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki.

[10] Bitcoin wiki. Protocol specification. https://en.bitcoin.it/wiki/Protocol_specification#tx.

[11] Bonneau, J., Clark, J., Felten, E., Kroll, J., Miller, A. and Narayanan, A. 2014. Mixcoin: Anonymity for Bitcoin with Accountable Mixes. In *Financial Cryptography*.

[12] Bonneau, J., Clark, J., Felten, E., Kroll, J., Miller, A. and Narayanan, A. 2014. On Decentralizing Prediction Markets and Order Books. In *Workshop on the Economics of Information Security*. https://www.cs.princeton.edu/~kroll/papers/weis14_prediction.pdf.

[13] Bonneau, J., Felten, E., Goldfeder, S., Kroll, J., and Narayanan, A. 2014. Securing Bitcoin wallets via threshold signatures. https://freedom-to-tinker.com/blog/stevenag/new-research-better-wallet-security-for-bitcoin/.

[14] Chiesa, A., Garman, C., Miers, I., Virza, M., Ben-Sasson, E., Green, M., and Tromer, E. 2014. Zerocash: Practical decentralized anonymous e-cash from bitcoin. In *IEEE Symposium on Security and Privacy (S&P)*.

[15] Decker, C. and Wattenhofer, R. 2013. Information Propagation in the Bitcoin Network. In *13th IEEE International Conference on Peer-to-Peer Computing (P2P)*.

[16] Garzik, J. 2012. Re: Bitcoin: the long game. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=128532.msg1370879#msg1370879.

[17] Hardin, G. 1968. The tragedy of the commons. *Science 162*, 1243–1248.

[18] Hearn, M. 2011. Bitcoin wiki: Contracts. https://en.bitcoin.it/wiki/Contracts.

[19] Hearn, M. 2012. Re: Network-enforced performance-bonded dominant assurance contracts. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=67255.msg785122#msg785122.

[20] Hearn, M. 2013. Micro-payment channels implementation now in bitcoinj. https://code.google.com/p/bitcoinj/wiki/WorkingWithMicropayments.

[21] Lerner, S. D. 2012. Re: Can governments spam or ddos the bitcoin network to death? *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=128548.msg1374959#msg1374959.

[22] Litecoin wiki. Comparison between litecoin and bitcoin. http://litecoin.info/User:Iddo/Comparison_between_Litecoin_and_Bitcoin#Faster_transaction_time.

[23] Maxwell, G. 2011a. *#bitcoin-dev chat*. http://bitcoinstats.com/irc/bitcoin-dev/logs/2011/11/05#l1320530158. freenode IRC.

[24] Maxwell, G. 2011b. Bitcoin wiki: Zero knowledge contingent payment. https://en.bitcoin.it/wiki/Zero_Knowledge_Contingent_Payment.

[25] Maxwell, G. 2011c. Re: Difficulty adjustment needs modifying. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=46498.msg556137#msg556137.

[26] Maxwell, G. 2011d. Deterministic wallets. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=19137.0, http://www.cs.technion.ac.il/~idddo/detwal.pdf.

[27] Maxwell, G. 2012. Re: Patching the bitcoin client to make it more anonymous. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=24784.msg991235#msg991235.

[28] Maxwell, G. 2013. Coinjoin: Bitcoin privacy for the real world. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=279249.0.

[29] Maxwell, G. 2013. CoinWitness. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=277389.0.

[30] Miers I., Garman C., Green M., and Rubin A. 2013. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE Security and Privacy (S&P)*, pages 397–411, 2013.

[31] Miller, A., Juels, A., Shi, E., Parno, B., and Katz, J. 2014. Permacoin: Repurposing bitcoin work for long-term data preservation. IEEE Symposium on Security and Privacy (S&P). http://cs.umd.edu/~amiller/permacoin.pdf.

[32] Mizrahi, A. 2013. Re: Colored coins lecture in Amsterdam conference. *bitcoinX mailing list*. https://groups.google.com/forum/#!searchin/bitcoinX/prediction$20market/bitcoinx/RG0F-EoT67o/ZlHueFpIABwJ.

[33] Nakamoto, S. 2008. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin.org*.

[34] Rosenfeld, M. 2012a. Analysis of hashrate-based double-spending. http://arxiv.org/abs/1402.2009.

[35] Rosenfeld, M. 2012b. Dynamic block frequency. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=79837.0;all.

[36] Rosenfeld, M. 2012c. Trustless, instant, off-the-chain Bitcoin payments. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=91732.0.

[37] Rosenfeld, M. 2013. Multi-PPS. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=281180.0.

[38] Schwartz, D. 2013. Re: Why ripple is a bad idea. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=142261.msg1515753#msg1515753.

[39] Tobey, J. 2011. Abe: block browser for bitcoin and similar currencies. https://github.com/jtobey/bitcoin-abe.

[40] User "cunicula", Rosenfeld, M., et al. 2011. Proof of stake brainstorming. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=37194.0.

[41] User "cunicula". 2012. Bribery: The double double spend. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=122291.0.

[42] User "QuantumMechanic" et al. 2011. Proof of stake instead of proof of work. *Bitcoin forum thread*. https://bitcointalk.org/index.php?topic=27787.0.

[43] Sompolinsky, Y. and Zohar, A. 2013. Accelerating bitcoin's transaction processing. http://www.cs.huji.ac.il/~avivz/pubs/13/btc_scalability_full.pdf.

A Performance evaluation

Here we provide benchmarks of our implementation of *follow-the-satoshi*, and discuss future improvements.

Our non-optimized reference implementation[11] is written in `Python` and interacts with the `SQLite` database of the `Abe` [39] block explorer. This reference code implements the *follow-the-satoshi* subroutine as described in Section 3, i.e. by starting from the coinbase [8] that contains the input satoshi and traversing forward according to the transactions that spend this satoshi, until reaching the unspent output. This implementation uses well-tested components and provides a sanity check, but it is also somewhat slow, in part because of the `SQL` queries that get invoked during each hop. Still, with a 2 gigabytes Bitcoin blockchain we obtain that on average each invocation of *follow-the-satoshi* does 60 hops and takes less than 0.1 seconds in total. The worst case time bound is $\mathcal{O}(K \log K)$ where K denotes the total number of transactions, because in each hop we query the `Abe` database via tree lookups that run in $\mathcal{O}(\log K)$ time.

Our `C++` implementation[12] of *follow-the-satoshi* eliminates all the intermediate traversal paths, and instead we maintain a database that keeps track of which saotshis are controlled by which outputs. This can be thought of as if each coinbase of newly minted coins is being partitioned into intervals, where each interval is associated with the stakeholder who currently controls those coins. With a 2 gigabytes Bitcoin blockchain, it takes on average about 1.5 microseconds per invocation of *follow-the-satoshi*, and about 0.2 seconds to process a newly solved block in order to update the intervals database. The worst case time bound of each *follow-the-satoshi* invocation is $\mathcal{O}(\log J) \leq \mathcal{O}(\log K)$ where J denotes the total number of intervals, because we reach the unspent output by searching through a tree of the intervals. The worst case time bound of updating the intervals database is $\mathcal{O}(J \log J)$, where the number of affected intervals can be proportional to J in the worst case, and $\mathcal{O}(\log J)$ is the cost of updating an entry. To avoid frequent database updates, we can have a tradeoff by using a hybrid algorithm that invokes a single interval lookup and then hops through the remaining transactions.

For an optimized implementation, we transform *follow-the-satoshi* into *UTXO-find-the-satoshi* (UFTS). Let U denote the number of unspent outputs that currently reside in the ledger. From a high-level view, suppose that each stakeholder keeps a sequence $(e_1, e_2, e_3, \ldots, e_U)$ of all the currently unspent outputs, sorted according to to their `<txid hash,output index>` pair [10]. To run UFTS, we derive from the empty PoW header (c.f. Section 3) a pseudorandom value v between 0 and the sum of all the unspent coin amounts, find

[11] http://www.cs.technion.ac.il/~idddo/test-fts.py
[12] https://github.com/killerstorm/blockparser/blob/master/cb/fts.cpp

the smallest i for which the sum of unspent coins in the subsequence $(e_1, e_2, e_3, \ldots, e_i)$ is greater than v, and select e_i. Note that UFTS picks a pseudorandom stakeholder in a uniform fashion, just like *follow-the-satoshi*.

We implement this sorted sequence as a binary search tree whose nodes are keyed by (a hash of) their `<txid hash,output index>` pair, and the value of each tree node is the sum of unspent coins in its subtree. We may presume that the height of this tree is $\mathcal{O}(\log U)$, either in a heuristic sense (assuming that the keys are pseudorandom), or by using a self-balancing tree. We insert or delete a tree node e_j in $\mathcal{O}(\log U)$ time by traversing from the root and updating the values (sums) of the nodes along the path to e_j. We invoke UFTS(v) in $\mathcal{O}(\log U)$ time by starting to traverse from the root, branching to the first child if its value is greater than v, otherwise branching to the second child if the difference between the current value and this child's value is less than v.

The efficiency gains of UFTS are indeed substantial. Consider a transaction with k inputs and one output $txout_0$. With our *follow-the-satoshi* "intervals" implementation, we need to update the identities of the k sub-intervals to $txout_0$, under the very likely assumption that these sub-intervals cannot be merged because they are not adjacent inside the same coinbase interval. With UFTS, we always merge the k elements into a single element. Hence, UFTS allows us to perform lookups in a data structure whose size is greatly reduced. Let us note that the Bitcoin protocol could be extended to include in each block the root hash of the current UTXO Merkle tree, to enable lite clients that wish to generate new blocks to be able to verify that an output is unspent. This means that the database that we need to maintain for UFTS can overlap with the UTXO set that the reference Bitcoin client already maintains, and in order to add support for UFTS the extra complexity of each database operation is minor. To efficiently handle re-organizations when an alternative chain outcompetes the current best chain, the network node should keep the recent previous blocks.

Regarding long-term scalability, succinct computational integrity proofs[13] can provide trust-free (w.h.p.) "checkpoints" for the UTXO set. This would be similar to the current developers-chosen checkpoints, except having each checkpoint commit to the hash of a UTXO set, with clients being able to verify quickly that the checkpoint is reached from the genesis block in accord with the protocol rules. Therefore, if we use UFTS instead of *follow-the-satoshi*, a stakeholder who operates a full network node will be able to prune the blockchain history.

[13] See e.g. http://www.scipr-lab.org/.