# A comparison of Solana and BlockfinBFT

This document discusses how Solana[1] consensus algorithm works and compares it to Storecoin's BlockfinBFT[2] consensus algorithm. All consensus algorithms have the following common properties[1] to address *consensus* in a distributed system consisting of a set of nodes that may not trust each other.

**Agreement** — All honest nodes agree on the same value. In blockchains, this means all honest nodes agree on the same block.
**Safety** — The agreement is achieved in the presence of adversaries such that different nodes will never decide on different values. This ensures consistency of the system.
**Liveness** — Liveness ensures that some *decision* will be taken *eventually*. The system as a whole will eventually make progress.

While all consensus algorithms have the common properties described above, they are designed with different assumptions and work in different network setups. The algorithms work well under the assumed conditions and if those conditions are not practical or can be easily broken, the promised benefits will not be realized.

This document compares the assumptions in Solana and BlockfinBFT consensus algorithms and how the algorithms behave when those assumptions are broken.

## Solana

Solana's design is based on an innovative concept called Proof-of-History (PoH), which addresses the *transaction ordering problem* in a peer-to-peer (p2p) distributed network. When transactions are received by different nodes in a p2p network at different times, how can they *order* these transactions at the network level? PoH addresses this problem without relying on timestamps (which can be faked) or requiring nodes to synchronize their clocks (which has external dependencies). Instead, it provides a mechanism for "verifiable passage of time", which allows any node in the system verify that *one event happened before or after another event*. Fig. 1 illustrates how PoH works.

[1] These definitions are not technical to keep them simple.

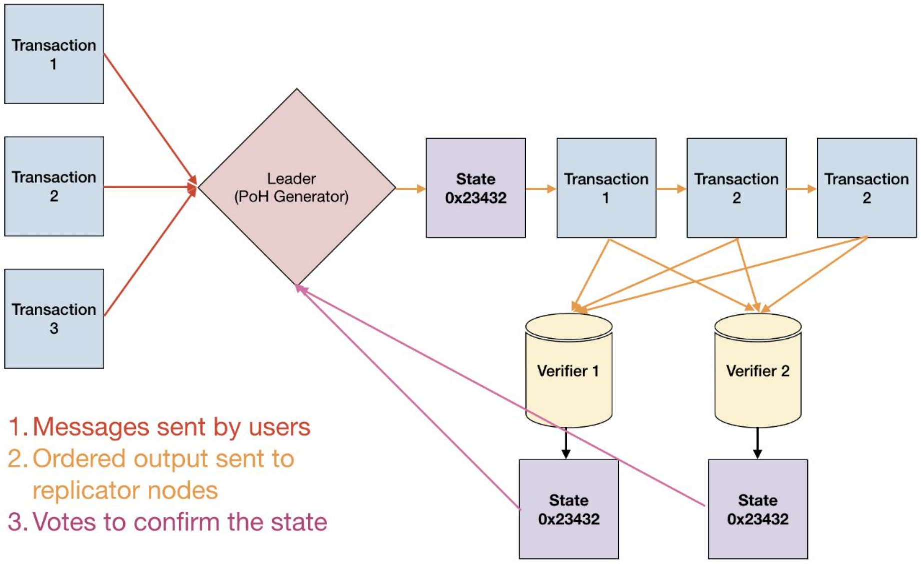
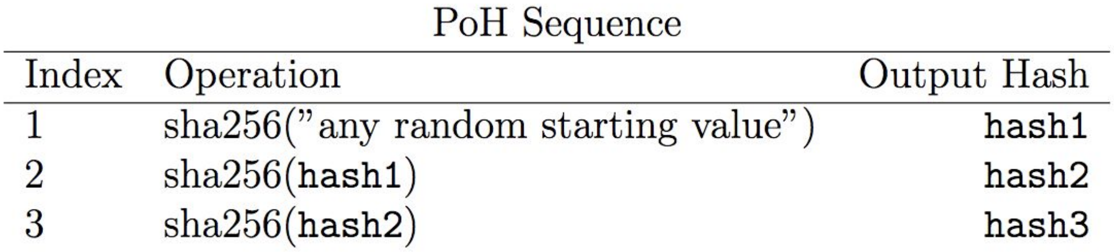


Fig. 1 — PoH in Solana. Illustration copied from [1]

Solana is a leader-based consensus algorithm where a leader orders transactions using PoH and relays them to replicator nodes, which verify the order and vote to confirm the state proposed by the leader. In this setup, the leader *creates* PoH and the replicator nodes *verify* PoH.

The details of how PoH is achieved are not important for the purpose of this analysis, but fig. 2 summarizes how it works. A random, but commonly agreed string is hashed with a hashing algorithm such as SHA256. The resulting hash is hashed again forming a sequence of hashes. In fig. 2 it can be verified that hash3 must have been created *after* hash2. One cannot guess or compute hash3 without starting from the random string and following the sequence of hashes. Thus, "verifiable passage of time" is achieved. If any event, such as a transaction, is attached with these hashes, we can thus verify that the transactions are in fact ordered along the same sequence.



PoH Sequence

| Index | Operation | Output Hash |
|---|---|---|
| 1 | sha256("any random starting value") | hash1 |
| 2 | sha256(hash1) | hash2 |
| 3 | sha256(hash2) | hash3 |

Fig. 2 — PoH is achieved with a sequence of hashes

Solana uses a PoS-based protocol built on top of PoH primitive. PoH generators bond their stakes to participate in the network. If PoH generators act maliciously, they are punished by slashing their bonds. PoS in Solana works as follows.

1. A leader is elected among bonded PoH generators based on the largest voting power, or highest public key address if there is a tie in the voting power.
2. The leader election takes place when a failure is detected with the current leader.
3. A super majority confirmation is required for the new leader.
4. The leader proposes the new state as illustrated in fig. 1 earlier and the verifiers verify the proposed state and vote for it.

The details beyond how PoH works are thin in Solana's white paper. We observe the following based on the details available in the white paper.

## Protocol assumptions and their effect in practical deployments

**Relying on clients for correctness** — It is possible for a malicious PoH generator to generate the hashes faster than other generators. This allows the malicious generator to order its transactions ahead of others. Solana addresses this by requiring clients to observe the sequence and embed what they consider as the latest hash in the transactions they send. This means, the protocol relies on the clients for correctness. Nowadays we have seen examples where clients exploit weaknesses in the protocol to their benefits, so a protocol relying on its clients for its correctness may be a concern.

**Leader is elected based on the voting power** — Leader rotation is not discussed in the white paper, so we assume that a leader, once elected based on the size of their stake, remains elected as the leader and hence PoH generator. This gives too much power to the leader and the leader becomes a known target for DDoS and spam attacks.

**An adversary can create multiple PoH generator accounts** — The protocol addresses Sybil attacks, but it doesn't address the problem where an adversary creates multiple genuine accounts with large voting powers. So, the secondary leader (which is used to take over the transactional processing duties) may also be controlled by the adversary.

**An adversary can create multiple verifier accounts** — This is an extension of the above scenario where the adversary controls sufficiently large number of verifiers. The protocol relies on verifiers' votes to slash a malicious leader, but verifiers' votes can be purchased by the adversary.

Tragedy of the commons — Verifiers have no incentives to verify the hashes generated by the leader, so the protocol addresses this issue by requiring PoH generator to inject an invalid hash at a random interval. Any voters for this hash should be slashed. However, an adversary can circumvent this by creating the necessary number of voters (verifiers) who collude with the leader.

**Censorship** — The white paper addresses censorship as follows. The highlighted text below makes many assumptions, which may not be practical in real work scenarios. This section also mixes censorship and DoS attacks, which are two separate issues. An adversary with a leader and sufficient number of verifiers can easily censor transactions. There is no guarantee that the new leader is not Byzantine. So, these descriptions are not credible enough to believe that they address the censorship or DDoS problems.

> Censorship or denial of service could occur when a ⅓ rd of the bond holders refuse to validate any sequences with new bonds. The protocol can defend against this form of attack by dynamically adjusting how fast bonds become stale. In the event of a denial of service, the larger partition will be designed to fork and censor the Byzantine bond holders. The larger network will recover as the Byzantine bonds become stale with time. The smaller Byzantine partition would not be able to move forward for a longer period of time. The algorithm would work as follows. A majority of the network would elect a new Leader. The Leader would then censor the Byzantine bond holders from participating. Proof of History generator would have to continue generating a sequence, to prove the passage of time, until enough Byzantine bonds have become stale so the bigger network has a super majority. The rate at which bonds become stale would be dynamically based on what percentage of bonds are active. So the Byzantine minority fork of the network would have to wait much longer than the majority fork to recover a super majority. Once a super majority has been established, slashing could be used to permanently punish the Byzantine bond holders.

**Verifiers may have a hard time synchronizing with the leader** — In real world scenarios, a verifier may not be able to respond within the stipulated timeout. So, what happens in such scenarios is not described.

> Each verifier is required to respond within a small timeout - 500ms for example. The timeout should be set low enough that a malicious verifier has a low probability of observing another verifiers vote and getting their votes into the stream fast enough

## Conclusions on Solana

1. PoH is a very innovative approach to order transactions without using timestamps or synchronized clocks.
2. However, their leader-based approach gives too much power to the leader because the leader is elected based on the stake size. While slashing is used to enforce honesty, it may not be

sufficient given that there is no proposal on leader rotation. So, a leader, once elected, may remain in that position as long as they maintain their uptime.

3. An adversary can successfully create a collusion between a leader and required number of verifiers to defeat the protocol. This is obviously expensive, but the rewards are worth the cost.
4. In steady state where there are no incoming transactions, PoH generators are still busy computing hashes. This overhead cannot be understated.
5. The protocol relies on clients for the correctness and honesty of PoH generators. This introduces additional attack vectors.
6. The protocol assumes *strong synchrony*, which is harder to achieve in a network with a large number of geographically distributed nodes. The white paper doesn't mention how verifiers converge in the event of synchrony failures.

## Storecoin's BlockfinBFT

We believe that transaction *ordering* in a block matters only when two or more transactions originate from ("Alice sends money to Bob and Charles") or terminate into ("Bob and Charles send money to Alice") the same account. If transactions are independent, their ordering is unnecessary and hence the ordering complexity can be completely eliminated. So, BlockfinBFT reduces the ordering problem to a *grouping* problem where the nodes only need to agree that the proposed block contains an agreed upon set of transactions. For example, consider a block proposal with 5 transactions, T1-T5. Different nodes may have received these transactions in different orders, so for example, node-A may have received them in

**T1 -> T2 -> T3 -> T4 -> T5**

where as node-B may have received these transactions in a completely different order

**T3 -> T1 -> T4 -> T5 -> T2.**

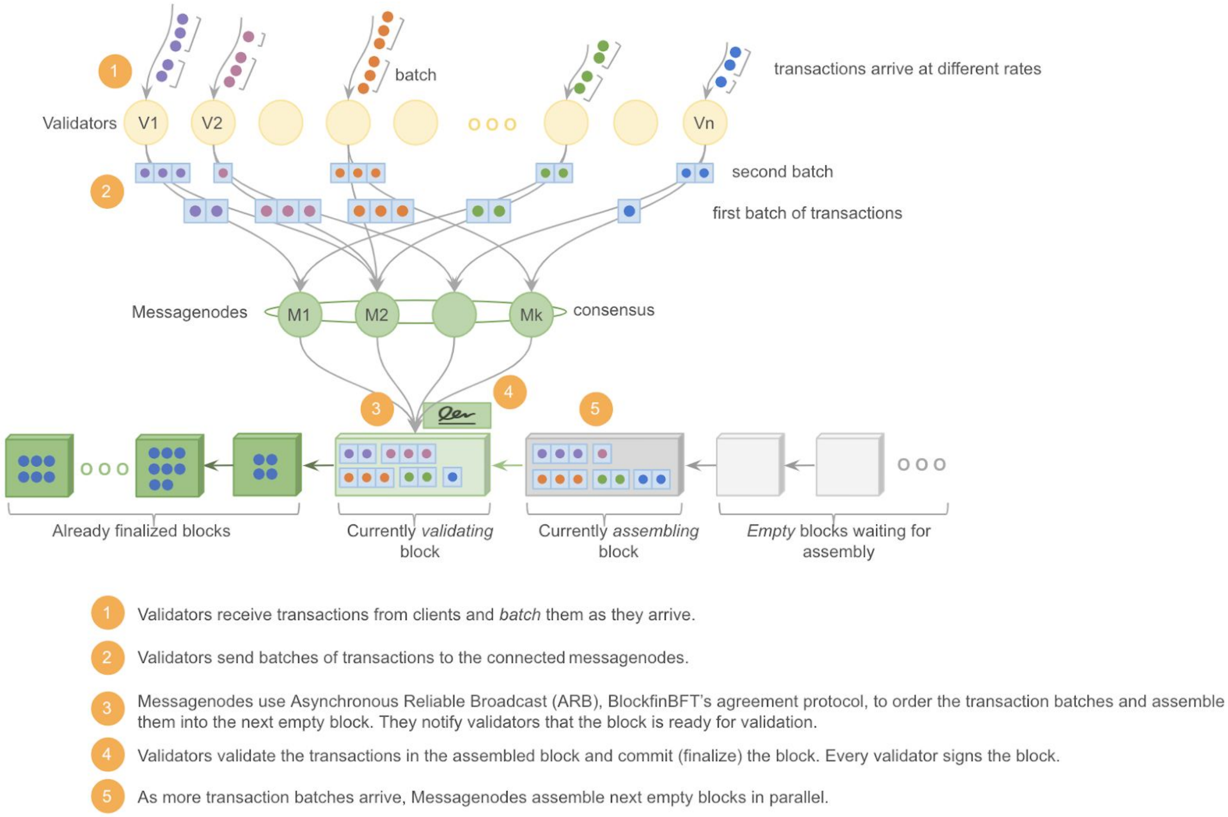
If nodes A and B must agree on the proposed block, it is sufficient to agree that:

1. they both have 5 transactions in the block and
2. the transactions are T1-T5, **but in any order**.

But what happens when the block contains transactions that originate from or terminate into the same account? In these cases the order matters for accurate accounting and to prevent double-spending.

The transaction order matters not during the consensus, but when they are executed in respective nodes, *after* they have agreed on the block's content. The ordering is done in respective nodes just before they execute transactions and update the states in the blockchain. This ordering is done based on the *nonce* field in the transaction, which is a monotonically increasing integer field. Nonce is incremented for every transaction originating from the same account.

With the need for transaction ordering removed from the consensus process, the need for PoH-like mechanism is unnecessary. Fig. 3 illustrates BlockfinBFT consensus process.


1
batch
transactions arrive at different rates
Validators
V1
V2
ooo
Vn
2
second batch
first batch of transactions
Messagenodes
M1
M2
Mk
consensus
3
4
5
ooo
ooo
Already finalized blocks
Currently *validating* block
Currently *assembling* block
*Empty* blocks waiting for assembly

1 Validators receive transactions from clients and *batch* them as they arrive.

2 Validators send batches of transactions to the connected messagenodes.

3 Messagenodes use Asynchronous Reliable Broadcast (ARB), BlockfinBFT's agreement protocol, to order the transaction batches and assemble them into the next empty block. They notify validators that the block is ready for validation.

4 Validators validate the transactions in the assembled block and commit (finalize) the block. Every validator signs the block.

5 As more transaction batches arrive, Messagenodes assemble next empty blocks in parallel.

Fig. 3 — BlockfinBFT consensus steps

At a high level, BlockfinBFT exhibits the following properties.

**One entity, one vote** — The votes are not weighed based on the size of the stake. One node casts a single vote during consensus rounds, no matter the size of their stakes. So, an adversary cannot deposit disproportionately large bond and earn a large percentage of voting power. This obviously addresses the Sybil attack.

**Leaderless consensus** — There are no leaders elected for block proposals. Every validator node is a leader for every block. They eventually converge on the same agreement on the transaction batches to be included in the next block. So, the protocol doesn't require complex randomization techniques like Algorand's Verifiable Random Function (VRF) to elect a block proposer. Leaderless consensus also makes the nodes less vulnerable to DDoS attacks because such attacks require targeting all the nodes.

**Equitable incentive structure** — The leaderlessness property means that the block rewards are not earned by a single leader. Instead, they are shared by all nodes who participated in building and validating the block. It is always possible that some nodes are offline during the consensus rounds for certain blocks and they don't earn their shares for the missed blocks, but otherwise, there is no *competition* to earn block rewards. So, there is no economic incentive to defeat the protocol rules.

**No strong synchrony assumptions** — The protocol implements no timeouts during consensus rounds. The protocol makes progress when more than ⅔ nodes sign the block in a round. Each node can proceed with its own speed. This means, if ⅓ or more nodes are unreliable, they can slow down the consensus process, but if the nodes are indeed unreliable (mostly due to unreliable network) requiring strong synchrony doesn't help anyways because that property can be hardly achieved.

**Collusion requires ⅓ or nodes to join hand** — Since the votes are not weighed by the size of the stake, a collusion requires ⅓ or more miners (they are called as *dWorkers* in Storecoin because Storecoin is a PoS-based blockchain, just like Solana and there is no mining involved) to join hands and approve malicious blocks. This then becomes a social engineering problem — where the adversaries look for like-minded miners to recruit and attack the network — rather than an economic problem — where an adversary can simply buy more stake to secure a large percentage of voting power. "One entity, one vote" makes it harder for a single entity create multiple miner accounts. Finally, the equitable economic incentive discourages cartel-forming because the possible gain from the collusion (mostly double-spend transactions) must really offset the potential revenue in block rewards for miners to agree on collusion.

No consensus algorithm is ever perfect and BlockfinBFT is no exception. Here are some of its drawbacks as we see.

1. Leaderless consensus implies that blocks are not created and finalized at predefined intervals. The speed at which blocks are created and finalized depends on the collective speed of the super majority of miners. If network conditions are favorable (closer to the *strong synchrony* model) the blocks are finalized quickly. Storecoin encourages better infrastructure for miners with two bonuses, which are paid to them in addition to their shares of the block rewards. One, an *uptime bonus* rewards them for maintaining a certain minimum uptime on a daily basis. Second, an *infrastructure bonus* rewards them for maintaining minimum hardware and network performance.

2. Finality is probabilistic. A block is said to be *finalized* when more than ⅔ nodes validate and sign it. The probability increases as more nodes sign the block and it reaches 1 when more than ⅔ nodes sign it. Given the asynchronous nature of the consensus rounds, time to finality cannot be deterministic.
3. The consensus rounds make progress when a sufficient number of signatures are collected for specific rounds. The signature overhead may be non-negligible if transactions trickle in at very low rate. In such cases, the number of transactions in a block may be very low with a constant overhead for signatures. While signature aggregation schemes such as [3] can be used to minimize this overhead, we have not seen an implementation that has low runtime overhead during consensus rounds. So, any such aggregation scheme may need to be applied as a post processing step rather than during the consensus process itself.

## Conclusions on BlockfinBFT

1. "One entity, one vote" model addresses Sybil attacks as well as an adversary acquiring disproportionate voting power.
2. Equitable incentive scheme removes incentives for cartel-forming.
3. Leaderless consensus process improves resilience to DDoS and spam attacks.
4. No timeouts or strong synchrony assumptions help us build a system that works in a real world setup. The protocol thus needs to make fewer assumptions.
5. On the downside, block intervals are not fixed and finality is probabilistic. Storecoin institutes additional bonuses to address these problems. The signature overhead per block may be justifiable only when the system is used to its full capacity.

# Final thoughts

Protocol designers typically ignore economic security when they design consensus algorithms. When money is involved, economic security is as important, if not more, as the security afforded by the technology. If people can misuse the system to their advantage, they will. Our research concludes that slashing alone is not a strong enough weapon to discourage cartel-forming. So, we built the economic security into the protocol with "one entity, one vote" and equitable incentive models. As for technology itself, fewer the assumptions made in the consensus layer, the better it works in real world scenarios. On this front, we believe that any leader-based approach has a long-term tendency towards centralization. This is especially true when the leader is chosen based on their economic power — the rich stay richer. A true decentralization doesn't need thousands of nodes in the network, but should discourage tendency towards centralization. Leaderless consensus approach in BlockfinBFT requires the participation of all nodes, which achieves this goal.

# References

1. https://solana.com/solana-whitepaper.pdf
2. https://research.storecoin.com/BlockfinBFT
3. https://crypto.stanford.edu/~dabo/pubs/papers/BLSmultisig.html