Consensus

Here is a new take on consensus using theBLS multi-signature scheme and random number generation using the[BLS single-signature scheme]. Some of the new blockchain protocols are using BLS multi-signature scheme for consensus, because it is fast, but they have serious problems with generating random numbers. This multi-signature scheme has a biasable result, as the leader can aggregate any subset of n

signatures, but since we won't use the result as a random seed and only to prove the consensus was reached, this is sufficient. Even though BLS is more time consuming on both signing and verification than Schnorr signatures, it reduces the communication rounds, so it actually reduces considerably the time spent in consensus.

The goal of the design is to have a coherent approach with security and liveness. One of the most important points of the system is having an un-biasable, 0-predictable random consensus group selection.

For clarification, the consensus operation flow withBLS multi-sig is explained below from a high level perspective.

1. Leader creates block with transactions and broadcasts this block to consensus group members

2. Each consensus group member validates the block, and if block is valid creates a BLS signature and sends this back to the leader.

3. Leader selects from among the received signatures a subset of at least n

and creates a bitmap for his selection, where B[i]=1

if the i

th member of consensus group was selected and B[i]=0

otherwise. Leader aggregates the signatures and broadcasts the block attaching the bitmap and signature to it (B, aggSig)

. It must also sign the end result, just to "seal" the configuration for (B, aggSig)

before broadcasting.

In this case we have several advantages in comparison to Belare-Neven multi-signature (a Schnorr multi-signature scheme), among which:

1. No single validator can cause the consensus to be aborted, except if that one is the leader, but there is no incentive to do this, the leader would only lose the fee => we gain real byzantine fault tolerance;

2. Reduce the consensus communication rounds from 5 to 2 which makes the consensus much faster (interactivity is much reduced) and again reduces possibility of abort due to connectivity, latency, etc => we gain performance.

Randomness source calculation

Here comes the new idea.

Signing blocks with BLS multi-signature means that the aggregated signature cannot be used as an unbiasable random number seed, since the leader has a high leverage on what the end result could be. For example, out of the group members signatures it can select any subset of size n

that is more advantageous to aggregate, this gives it a combination of cSize

taken n

options to choose from.

In this case we can try to find another source of randomness that we can create with little effort and has the properties we mentioned before:

1. It needs to be unbiasable;

2. It needs to be unpredictable.

3. It needs to be fast and reliable.

4. It needs to be provable.

BLS single signature as candidate

BLS single signature (a summary here) seems to share the same property with deterministic k generation ECDSA, in that signing the same message with the same private key always produces the same result.

At the same time the scheme looks very simple:

Given private key sk

, message m

, and a hashing function H

that hashes to points on G1

:

In this case, there is no biasable parameter that could be changed such that the signature would still be verifiable but would give multiple options to the signer on the end result.

How we could use this with the proposed consensus model:

1. New consensus group is selected using a hash of the randomness source taken from the previous block header (a BLS signature, or in case of the genesis block a known seed).
2. Chosen leader signs the previous randomness source with BLS single signature to generate the new randomness source, creates a block with transactions, adds the new randomness source in the block header and broadcasts this block to consensus group members;
3. Each member validates the block, also validating that the new randomness source is a signature verifiable with the leader's public key on the old randomness source. If both are valid, it creates a BLS signature on the proposed block and sends this back to leader;
4. Leader selects from among the received signatures a subset of at least n

and creates a bitmap for his selection, where B[i]=1

if the i

th member of consensus group was selected and B[i]=0

otherwise. Leader aggregates the signatures and attaches the bitmap and signature to the block. It must also sign the end result, just to "seal" the configuration for (B[], aggSig)

before propagating the resulted block through gossip inside the shard.

The evolution of the randomness source in each round can be seen as an unbiasable and verifiable blockchain, where each new randomness source can be linked to the previous randomness source. So we created a blockchain of randomness numbers, which is independent of transactions, messages, blocks or anything. Furthermore, the chain unpredictable.

The randomness source is known at most 4 seconds before the next group is selected, but it is unchangeable. It is like a VRF chain without specialized hardware.

Conclusion

The newly proposed consensus model with the BLS single signature scheme for randomness, is the chosen solution for improving the liveness of the consensus and generating unbiased random numbers, with a low degree of predictability (one round). In order to have the random seed uniformly distributed we can apply on top a hashing function and use instead of the BLS signature the hash of this signature to select the next consensus group.

This would mean that we have a consensus that uses aggregated BLS multi-signature as block signature to prove the consensus was reached, and single BLS signature from the leader of the consensus over the last randomness source providing the next randomness source. The transactions still uses Schnorr signing, as it is much faster to verify than BLS (one order of magnitude faster).

What we gain is a faster consensus, with 2 rounds of communication instead of 5, of complexity O(n)

instead of $O(n^2)$

since leader needs to broadcast to the other members and the other members only need to send back to leader their replies. This means that we can increase even further the consensus group size (e.g 61 members should be easy) and as effect improving the shard security, as probability for malicious majority falls to $\sim 1/10^9$

per round with 61 members in the consensus group.

The consensus also becomes more fault tolerant, improving liveness, since it does not matter which signatures the leader chooses as long as there are n

of them.

The randomness source becomes unbiasable, either by leader or any other consensus member, since only the leader of a consensus can provide the next randomness source, but this result is fixed and only depends on the leader’s private key and the previous randomness source.

Because the consensus becomes faster with fewer interactivity rounds, we have more options for the economic model. For example we could increase the rewards per validator per block with number of signatures a leader aggregates in the BLS block multi-signature. This would have been hard with previous solution, but now becomes easy.

Any thoughts on the randomness number generation ?

More here: Change in Consensus and Randomness Source