TLDR

: We suggest using BLS signature aggregation as a pragmatic medium-term solution to the signature verification bottleneck of sharding and Casper, later replacing it with STARK-based aggregation for quantum security.

Background

Signature verification is the major bottleneck for both sharding and Casper. Assuming 32 ETH deposits (relatively small deposits to encourage participation and decentralisation) and ~10,000,000 ETH at stake we get ~312,500 validators. Casper finality requires 2/3 of the validators to vote, and with ECDSA signatures a single on-chain Casper cycle would take ~1 day.

For sharding, the signature verification bottleneck forces us to compromise:

1. Full committee safety and liveness

: We use small committee sizes (e.g. 135) counterbalanced with a high threshold above 50% (e.g. 66.6% threshold with 90-of-135) to maintain an acceptable honesty assumption. With large committees (e.g. of size 1000) we could have an optimal 50% threshold for improved liveness, and we would reduce our honesty assumption for improved safety.

1. Windback strength

: Instead of fully verifying committee votes in the windback, we compromise by requiring that validators only verify CASs. Because CASs are trivial to forge (at the cost of losing 32 ETH deposits) this weakens the strength of notarisation for collations after the latest 0-skip (full committee vote), opening the possibility for short term attacks.

1. Crosslinking frequency

: We limit the frequency at which crosslinks get included in the beacon chain to about one crosslink per shard per hour, yielding slow crosslinking. Crosslink finality is also delayed which affects the usability of cross-shard transactions.

1. Number of shards

: The maximum number of shards is limited by the signature overhead the beacon chain has to bear for crosslinking. With ECDSA signatures crosslinking for 100 shards is enough to saturate the beacon chain.

(With signature abstraction individual signatures can be an order of magnitude more expensive to process, worsening the bottleneck compared to ECDSA.)

One strategy to relieve the signature verification bottleneck is signature aggregation. Two aggregation techniques have been developed in-house but neither is fully satisfactory:

1. Cryptoeconomic aggregate signatures

: CASs work well for securing non-mission-critical infrastructure (e.g. signalling to light-clients about collation validity and availability near the shard heads) but they are too slow to verify for crosslinking purposes.

1. Lamport multisignatures

: Lamport multisignatures work well for multisignatures (sharding crosslinks and Casper checkpoints) but they are made of weak 5-bit “proto-signatures” which make them inappropriate for use with CASes and proofs of custody.

Two other options we have considered for aggregation are BLS signatures and STARKs. Up until recently both were not viable. Specifically, while BLS signatures can shrink the size of signatures, they would still require $n+1$

relatively expensive pairings to verify n

aggregated signatures. Furthermore, BLS signatures are not quantum-secure. As for STARKs, their 3-4 MB size made them unwieldy for the foreseeable future.

Coincidentally, and in a stroke of good fortune, there have been two recent cryptographic breakthroughs:

1. BLS aggregate signatures

: It is now possible to verify n

aggregated signatures on the same message m

with just 2 pairings instead of $n+1$

thanks to a paper published a week ago.

1. STARKs

: Eli and his team from StarkWare have been able to reduce the size of STARKs to 80kB.

These two breakthroughs combined point to an attractive roadmap for signature aggregation:

- Short and medium term

: As argued below BLS aggregation works extremely well for sharding and Casper (apart from the lack of quantum security). They are a pragmatic solution for the short and medium term (say, the next 5 years).

- Long term

: STARKs are the promising generic end-game technology which can be used to build a quantum-secure equivalent to BLS aggregation. The 80kB-size breakthrough suggests that STARK aggregation may be used as a drop-in replacement to BLS aggregation within a reasonable time frame (say, the next 5 years).

BLS aggregation design

Before digging into BLS aggregation performance and example use cases we recap the simple and elegant design (the notation differs from the paper):

- Setup

: Let $e : G_1 \times G_2 \to G_t$

be an efficiently computable non-degenerate pairing where G_1, G_2, G_t

are groups of prime order p

. Let g

be a generator of G_2

.

- Key generation

: Every validator picks a secret key $sk\in\mathbb{Z}_p$

and submits g^{sk}

to the beacon chain. The beacon chain then computes and stores the corresponding public key $pk=(g^{sk})^{\textsf{H}(g^{sk})}$

where $\textsf{H}$

is a hash function mapping to $\mathbb{Z}_p$

.

- Signing

: To sign a message m

the validator with public key pk

produces the signature $s = \tilde{\textsf{H}}(m)^{sk\cdot \textsf{H}(g^{sk})}$

where $\tilde{\textsf{H}}$

is a hash function mapping to G_1

.

- Aggregation

: The aggregate signature $\sigma=\prod_{i=1}^{n}s_i$

is the product of individual signatures s_i

. The aggregate public key $\pi = \prod_{i=1}^{n}pk_i$

is the product of individual public keys pk_i

.

- Verification

: Signature verification is the two-pairing check $e(\sigma, g) \stackrel{?}{=} e(\textsf{H}(m), \pi)$

.

Notice that aggregation is non-interactive (i.e. signatures can be aggregated by anyone after broadcast) and is incremental (i.e. incorporating a new signature s_{n+1}

into an aggregate signature σ

can be done with multiplication).

BLS aggregation performance

As suggested in the BLS paper we use the 128-bit securityBLS12-381 curve. This curve will be used by Zcash for its next-generation SNARKs, and by Chia (see here) to compress signatures for its blockchain.

The curve is asymmetric (i.e. $G_1 \ne G_2$

) with one of the two groups G_1, G_2

being big-and-slow, and the other small-and-fast. For performance Zcash uses G_1

as the small-and-fast group, whereas Chia uses G_1

as the big-and-slow group (see here). We follow Chia's direction by having G_1

be big-and-slow.

Concrete overheads (numbers from the Zcash implementation optimised for safety; Chia is working on speed optimisations):

- Size

: Elements of G_1

have size 96 bytes and elements of G_2

have size 48 bytes (actually 381 bits, hence the name) .

- Multiplication

: Multiplication in G_1

takes 4,500ns and multiplication in G_2

takes 1,350ns.

- Pairing

: A pairing takes 2,700,000ns.

Example 1: Casper finality loop

Let's assume 2/3 of the 312,500 validators voted on the same Casper checkpoint. The corresponding aggregate signature would consist of 312,500 bits (39,062 bytes) describing the subset of validators that voted. The aggregate signature proper would add another 96 bytes for a total under 40kB. For on-chain verification, it would take 2/3 * 312,500 * 1,350ns = 281.25ms to reconstruct the aggregate public key, plus 2 * 2.7ms for the two pairings, for a total under 300ms on a single core. With quad-core parallelisation it would be ~75ms.

The bottleneck for Casper signature processing is no longer signature verification (the beacon chain can easily verify a full Casper vote every minute). Instead, the question is how fast can 2/3 * 312,500 messages (each of size ~300 bytes including Ethernet, IP and TCP headers) be aggregated offchain?

One natural direction is "sharded aggregation" where beacon chain leaders are only responsible for aggregating signature in the shard for which they are a proposer. (Each shard has 3,125 infrequently shuffled proposers.) We can reuse the shard gossip channels, and limiting throughput to 4 messages per second allows us to aggregate all messages in about 9 minutes.

Example 2: Crosslinks

Let's assume that at every beacon chain block (every ~5 seconds) a 500-of-1000 committee crosslink is added. In total a crosslink would be ~300 bytes (1000 bits to describe which validators voted, plus 96 bytes for the aggregate signature, plus 32 bytes for the chunks root, plus overhead) and would take at worst 1000 * 1,350ns + 2 * 2.7ms = 6.75ms to verify.

This is a small load on the beacon chain every 5 seconds. Because each shard would get crosslinked on average once per 100 blocks, crosslinking would happen roughly every 9 minutes.

Conclusion

With BLS signature aggregation we may be able to address the signature verification bottleneck. This would allow for 10-minute onchain Casper finality loops, 500-of-1000 committees, stronger windback validation, 10-minute crosslinking, and more shards (possibly up to 1,000 shards).

If and when quantum computers threaten BLS cryptography the plan is to do a drop-in replacement using STARKs.