Pragmatic signature aggregation with BLS

signature-aggregation

JustinDrake May 31

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.
2. **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.
3. **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.
4. **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.
2. **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.
2. **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 $pk = (g^{sk})^{\mathsf{H}(g^{sk})}$ where 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 $s = \tilde{\mathsf{H}}(m)^{sk \cdot \mathsf{H}(g^{sk})}$ where $\tilde{\mathsf{H}}$ 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 $e(\sigma, g) \stackrel{?}{=} e(\mathsf{H}(m), \pi)$.

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 security **BLS12-381 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. $G_1 \neq G_2$) 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.

A proposal for structuring committees, cross-links, etc

Which BLS curve/DKG algorithm is going to be used for Casper?

Committee-based sharded Casper

1-bit aggregation-friendly custody bonds

---

ldct **May 31**

Sorry, I haven't been reading the posts on the new sharding designs, so I have a question from the point of view of Casper only. What is a "beacon chain" in the context of Casper-FFG?

---

kladkogex **May 31**

As a side comment, one should be careful about using anything that comes out of Stanford since Stanford has a policy of patenting things. The original BLS scheme was patented.

Another question I have is why is it an aggregate signature and not a threshold signature? My understanding is that for an aggregate signature all parties need to participate, so a malicious validator can simply withold its signature

---

JustinDrake **May 31**

> ldct:
>
> What is a “beacon chain” in the context of Casper-FFG?

Good question 🙂 It’s the same beacon chain as for sharding. We are considering merging Casper and sharding such that the Casper FFG system logic and the sharding system logic both run on the beacon chain instead of on the legacy main chain.

For validators to make 32 ETH deposits the legacy chain would have a one-way burn contract. Withdrawals of deposits and rewards would remain disabled until we ship the EVM 2.0 in the shards. Advantages of this new approach include:

- **Segregation**: There would be a clean segregation between system logic and application logic. This makes it easier to upgrade the system logic without upsetting dapps. It also means that system contracts won’t have to compete with dapps on gas.
- **Future-proofing**: Eventually the EVM as we know it (EVM 1.0) will be deprecated and replaced by EVM 2.0 in the shards, at which point the beacon chain will become the root chain. As such, we want to avoid writing long-term system logic in EVM 1.0. Also full PoS would likely get shipped earlier.
- **Unity**: There would be a single version of proof-of-stake instead of one for Casper and one for sharding. This allows for capital reuse (all validators both Casper and sharding validators), a unified Ethereum 2.0 design, and closer cooperation between the Casper and sharding teams.
- **Simplicity**: Classes of issues go away, such as dealing with gas and writing code in Solidity. While withdrawals are disabled the manager contract on the beacon chain will be “unhackable”, and much easier to write and deploy.

> kladkogex:
>
> Stanford has a policy of patenting things. The original BLS scheme was patented.

Good point—I’ll look into it. Do you have a link to the patent for the original BLS scheme?

> kladkogex:
>
> why is it an aggregate signature and not a threshold signature?

We can use an aggregate signature, it just means that a fresh public key needs to computed for every new subset of signing validators (as opposed to always reusing the same public key). That’s OK because computing public keys only involves fast multiplications and is highly parallelisable.

The new BLS paper does also provide a threshold signature scheme but it requires a one-time interactive setup to build the public key. This setup doesn't scale well and doesn't easily allow for changes to the validator set.

Registrations, shard count and shuffling

---

kladkogex **May 31**

I think here is the original patent that cites threshold encryption among other things related to IBE

https://patents.google.com/patent/US7113594B2/en?q=boneh&q=threshold&oq=boneh+threshold

An entire company (Voltage Security) was based on this patent. I guess they are not enforcing is so much anymore, although Aug 13 2001 is less than 20 years in the past.

---

jamesray1 **Jun 1**

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

Got a reference for that? I can't find any info searching online.

---

adamluc **Jun 14**

Curious for some clarification, how does the legacy main chain (current chain) relate to the proposed beacon chain?

JustinDrake **Jun 15**

> kladkogex:
>
> here is the original patent

Dan Boneh didn't file BLS patents, and is not aware of patents on BLS.

> jamesray1:
>
> Got a reference for that? I can't find any info searching online.

Sorry I don't think it's online yet. The info came from discussions with Eli Ben-Sasson. As I understand 80bK is the size StarkWare got for the equivalent of a single Zcash transaction. There are a couple of factors that may increase the size to the 100kB-200kB range for us:

1. The size of a STARK scales roughly logarithmically with the length of the computation, and our aggregate signatures do more computation
2. There's a tradeoff between size and prover/verifier time, and it may be favourable to increase proof sizes a bit.

> adamluc:
>
> how does the legacy main chain (current chain) relate to the proposed beacon chain?

The beacon chain is a sidechain to legacy chain in the sense that:

1. Beacon chain blocks contain a reference to a legacy chain block.
2. If the legacy chain reorgs then the beacon chain must reorg accordingly.

At some point in the future the legacy chain can become a sidechain to the beacon chain.

burdges **Jun 18**

In fact, the blog post does use $pk_1, \ldots, pk_n$ when it writes $t_1, \ldots, t_n \leftarrow H(pk_1, \ldots, pk_n)$. It just folds the pk_i into the list and distinguishes by output stream location.

I'm surprised about there being an attack myself, but maybe I have not read the paper closely enough. Afaik, there is no way to avoid the public key exponentiation anyways since the versifier must check that too anyways.

Is there a reference for this proof-of-possession thing? It's just adding the BLS signature on themselves? I'd need to think about the attack when using the same hash function.

JustinDrake **Jun 18**

> the blog post does use $pk_1, \ldots, pk_n$ when it writes $t_1, \ldots, t_n \leftarrow H(pk_1, \ldots, pk_n)$

Oh right! I miss-read that 🙂

> Afaik, there is no way to avoid the public key exponentiation anyways since the versifier must check that too anyways.

Yes but it would be a one-time cost, and the verification could have been done at registration by the blockchain at no cost to verifiers.

> Is there a reference for this proof-of-possession thing? It's just adding the BLS signature on themselves? I'd need to think about the attack when using the same hash function.

See **this paper** . On page 4 it states: "We show that the standardized POP mechanism described above, when applied to these schemes, does not lead to secure multisignatures. Both schemes fall to rogue-key attacks despite the use of the standardized POPs. We present a straightforward and natural fix for this problem: simply use separate hash functions for POPs and multisignatures."

---

burdges **Jun 18**

Thanks for the reference!

> JustinDrake:
>
> Yes but it would be a one-time cost, and the verification could have been done at registration by the blockchain at no cost to verifiers.

I think this depends what you means by blockchain: If you mean the whole network, then yes but if accounts are single use then amortized this gives the same cost. If you means some smallish set, then no because an adversary could corrupt that entire set, and then submit false transactions.

---

burdges **Jun 18**

I figured it out, probably. We already can aggregate the proofs of possession because they are on different messages, so this should all be fine in the end.

Aggregation over different messages is not as efficient because you need one pairing per message, but that's better than two pairings per message.

Actually, one exponentiation per key for Dan's scheme should be dramatically faster though.

I'm also wondering if the exponents need to be big. In Dan's paper, they must be big because of the reduction to co-CDH, but intuitively 128 bits sounds sufficient so one wonders.

JustinDrake **Jun 18**

I don't understand the concern of your last two replies. Are you worried about the costs of registering a new BLS public key?

> burdges:
>
> if accounts are single use

Accounts are not at all single use. Keep in mind that the deregistration period of a validator will be ~4 months, and that every validator is invited to make a signature every ~5 second for attestations. So validator accounts will likely be making millions of BLS signatures in their life time with the same public key.

> burdges:

> If you means some smallish set, then no because an adversary could corrupt that entire set, and then submit false transactions.

I don't understand this sentence. 🙂 Set of what? What false transactions?

> burdges:
>
> We already can aggregate the proofs of possession because they are on different messages

Right, we could aggregate proofs of possession for ~2x reduced verification costs. But is the extra complexity worth it? Without aggregation each proof of possession takes ~5ms to verify so even assuming conservatively 100,000 registrations per month that's ~4 minutes of verification time saved per month.

> burdges:
>
> one exponentiation per key for Dan's scheme should be dramatically faster

The one-time proof of possession might be slower to verify than an exponential, but then every single signature (of which there are millions per public key) would be faster to verify.

---

burdges **Jun 18**

> JustinDrake:
>
> Are you worried about the costs of registering a new BLS public key?

I had not quite understood if registration was even the right model.

> JustinDrake:
>
> Keep in mind that the deregistration period of a validator will be ~4 months, and that every validator is invited to make a signature every ~5 second for attestations. So validator accounts will likely be making millions of BLS signatures in their life time with the same public key.

I see. We're not even talking about account keys, but validator's signatures. In that case, there isn't so much difference between the threat models anyways, so probably fine.

> JustinDrake:
>
> I don't understand this sentence. 🙂 Set of what? What false transactions?

I'm asking about corrupting the entire validator set, or maybe just 2/3rds, entering rogue keys for large accounts that rarely move, and much later stealing the balances form the target accounts. It's kinda the long-term double spend attacks, but with a simpler payoff and an arbitrary delay that likely increases its viability.

You're not talking about aggregating transaction signatures though, and doing so must deal with different messages anyways, while only validator signatures cover the same message.

---

vbuterin **Jun 20**

> burdges:
>
> I'm asking about corrupting the entire validator set, or maybe just 2/3rds, entering rogue keys for large accounts that rarely move, and much later stealing the balances form the target accounts. It's kinda the long-term double spend attacks, but with a simpler payoff and an arbitrary delay that likely increases its viability.

I'm confused here. How is entering rogue keys for other accounts even possible if you have to make a proof of possession at time of registration?

---

burdges **Jun 20**

> vbuterin:
>
> I'm confused here. How is entering rogue keys for other accounts even possible if you have to make a proof of possession at time of registration?

Rogue keys are not possible under the assumption that registration happened correctly. I'm pointing out that assumption can be violated more easily than standard cryptographic assumptions. In particular, a correct registration assumption might hold for one proof-of-stake scheme but cause problems for another one that handles the economic threats differently.

---

vbuterin **Jun 20**

Ah, I see. I think in general registration is an unavoidable part of all of the kinds of deposit-based PoS algorithms we are using, because a signature is not even valid in a beacon chain unless the validator that made the signature has already sent a deposit transaction and been inducted into that beacon chain. So it's totally ok to assume that registration happened correctly in our case.