TL;DR:

PEPC-DVT can keep the block secret while obtaining signatures from the DVT network, thanks to BLS Blinded Multi-Signatures.

## Glossary

- Distributed Validator

: A validator with a private key divided into multiple shares.

- Co-signers/Co-validators

: Holders of the shares constituting a distributed validator's key. They are the nodes that make up the validator's DVT network.

Reference

: The concepts discussed are based on Foteini Baldimitsi's presentation onBLS Multi-Signatures for Blockchain Applications.

## Current PEPC-DVT Challenges

In the existing PEPC-DVT framework, when a user wants to obtain the validator's signature, the block must be disclosed to all of the validator's co-signers.

This disclosure can lead to

- Information asymmetry, where co-validators might misuse early block knowledge.
- Co-validators potentially basing their signature on block content, which is not ideal — the signature should only depend on commitment fulfillment.
- It being impossible to implement a PEPCified MEV-boost because the relay would have to reveal the payload to all co-signers to gather their signatures.

Fortunately, not everything is lost: we can leverage BLS blinded multi-signatures to keep the block confidential while having co-signers provide their share of the signature. This can be done without co-signers taking the risk of signing a commitment-invalid block.

## Introduction to Blind Signatures

Blind signatures are cryptographic tools that enable message signing without revealing its content. A successful blind signature should ensure

- Unlinkability

: The commitment and its unblinded signature should remain uncorrelated.

- One-More Unforgeability

: Restricts adversaries from creating extra valid signatures beyond their allowance.

The process involves a user creating a blinded commitment of the payload, which the signer (in the case of only one) then signs using their private key. The user can later unblind this signature to obtain a signature identical to the one if the signer had signed an unblinded payload.

### Multi-Signature Blind Signatures

For PEPC-DVT, the scenario involves multiple signers. The desired properties include

- Correctness
- Multi-one-more-unforgeability
- Blindness

In this setup, each signer signs the same base payload. After collecting the blind signatures, the user aggregates them into a single multi-signature.

Note that the user might send distinct commitments to each signer, as is the case in the diagram above.

## Making PEPC-DVT private

Leveraging BLS multi-signatures, we can integrate blind signatures into PEPC-DVT. In BLS, the message undergoes hashing to a curve point. A blinded commitment is derived by multiplying this point with a generator point raised to a random number. The signature on this commitment is then calculated using the signer's secret key. To unblind a signature, it's multiplied by the signer's public key raised to the negative power of the initial random number (i.e., the same randomness used to blind the payload is used to unblind

it).

As before, we represent each signer with a unique public and private key pair. For the key aggregation, we multiply every public key together to get $apk=\Pi pk_i^{a_i}$

, where

$a_i=H_1({pk_1,\cdots,pk_n},pk_i)$

In other words, we hash together the set of all signers and the public key of the corresponding signer for $a_i$

. To obtain the aggregate public key, we multiply them all together to obtain $apk$

.

The user creates commitments for each signer and exchanges these commitments for signatures. These signatures are then aggregated using the same pattern as for aggregating public keys. For some random $r_i$

., the commitment to send to signer $i$

is given by

$C_i=H_2(\text{payload})g^{r_i}$

The user exchanges $C_i$

with every signer to obtain the blinded signature for each:

$\sigma_i'=C_i^{sk_i}$

To unblind it, we have

$\sigma_i=\sigma'_ipk_i^{-r_i}$

We then aggregate them to obtain the aggregate signature

$\sigma=\Pi \sigma_i^{a_i}$

Notice how this aggregate signature has the same pattern as the aggregate public key $apk=\Pi pk_i^{a_i}$

.

The verification is done as it normally is for BLS. That is, we check that

$e(\sigma,g)=e(H_2(\text{payload}), apk)$

which is two pairings checks because we are considering only the aggregated signature and the aggregated public key.

In summary,

## Verifying Commitment Fulfillment

Blind signatures introduce a challenge: verifying if a block meets commitments. Since signers can't view the payload's content, they can't verify its commitment fulfillment. To address this, we propose three methods.

### 1. Zero-knowledge Proofs

By running the commitment-verification process as a zkVM's guest program, commitment validation can be achieved. For instance, using RISC Zero's zkEVM sample and Reth, a call to Emily in the guest program of the zkVM can be made, where the payload is passed as the input. However, the computational cost of the hash function in BLS, which would need to be carried out in the guest program, can introduce latency.

### 2. Relay Mechanism

A relay can act as an intermediary. The user discloses the payload only to the relay, which then checks the payload's commitment fulfillment, computes the blinded commitments, and sends them to the signers.

Potential risks include the relay failing to function or maliciously revealing the payload. To mitigate these risks, multiple relays can be used in parallel by the user, or the relay can be run in an SGX environment.

### Aside

: PEPCified MEV-boost: Preventing unbundling of the block

The user may be the builder, who sends the payload to the relay so that the relay blinds it and obtains the signature from the validator's signers. Since signers only see the blinded payload, there's no risk of MEV stealing and the builder has the uncertainty that, as long as the relay is honest, the payload is never exposed to anyone else. This aligns with the principles of MEV-boost. Note that the blinding and sharing of the payload with the DVT network could be done without the relay. However, for consistency with the existing MEV-boost design, I used the same approach.

### 3. Stake-based Approach

Users can deposit a stake that's forfeited if the payload turns out to violate commitments. This ensures that users are financially incentivized to only share commitment-valid content.

## Closing thoughts

The integration of BLS Blinded Multi-Signatures into PEPC-DVT offers a privacy-centric approach without compromising the protocol's integrity. By considering various methods for verifying commitment fulfillment, including several that don't rely on intermediaries like relays, we can ensure both privacy and commitment adherence, paving the way for a more secure and private environment for general-purpose commitments.

Thanks to Dan Marzec and Barnabé Monnot whose questions pushed me to think in this direction.