Hyperledger Besu CIP

Hart Montgomery *

1 Introduction

In this document, we present the Besu CIP proposal in our own words. We first outline some core background material that is essential for understanding the proposal. Then we delinineate the terms of the proposal.

2 Background

We start by discussing some background material.

2.1 Terminology

We begin by providing some informal definitions of terms from the Ethereum ecosystem that will be used in this document. We emphasize that some of these terms have been redefined by the community over the lifespan of Ethereum, so we encourage readers who are unfamiliar with the current state of Ethereum to take note.

Eth1 vs Eth2. Eth1 defines the "execution layer" of Ethereum, while Eth2 defines the "consensus layer." This was recently formalized in a blog post from the Ethereum Foundation [Eth22d]. Note that this differs from previous notation. As an example, Hyperleger Besu will live (mostly) at the Eth1 layer since its primary focus is as an Ethereum execution client. On the other hand, Consensys's Teku [Con22] lives at the Eth2 layer.

2.2 Proof of Stake Consenus

A *consensus protocol* is a necessary feature of every blockchain because it protects the blockchain against potentially malicious adversaries. The Ethereum Bellatrix upgrade [Eth22a] changes the core consensus algorithm of Ethereum from proof of work to proof of stake. It would take an entire textbook to explain proof of stake consensus (we recommend [Shi20] as a general reference on modern consensus algorithms), so we do not attempt that here. We emphasize, though, that this proposal is written in a way that requires some understanding of proof of stake consensus in order to understand the rationale behind some of the design decisions. The goal of this proposal is to incentivize the development of an Ethereum mainnet client (Besu) and ensure that some validators run Besu on the Ethereum mainnet.

Informally speaking, proof of stake consensus involves participants on the network "voting" on transactions. Each participant's vote is weighted by the amount of stake they hold. If a participant misbehaves, they may lose their stake in a process called *slashing*. There is a huge amount of literature on proof of stake consensus [BG17, KRDO17, DPS19] and we encourage interested readers to peruse this literature for detailed explanations. We note that the Ethereum proof of stake specification is defined in a github repository [Eth22b].

*The Hyperledger Foundation.

2.3 Keys

In cryptocurrencies, ownership is typically represented by the ownership of signing keys. Ethereum is no exception. In our case, there are three primary sets of keys with which we need to deal.

Withdrawal Keys. The *withdrawal keys*, referred to in the specification as *withdrawal_credentials*, control the ability to withraw staked Ethereum, as well as receive validation rewards. These are best described in the aforementioned specification [Eth22b].

Currently there are two forms of withdrawal credentials supported:

- BLS signature [BLS01] key pairs over the curve BLS12-318 [BLS02, Bow17]. Note that even though they have the same acronym, BLS signatures are a cryptographic signature scheme (supporting aggregate signatures), and BLS curves are pairing-friendly elliptic curves on which BLS signatures can be built. We will avoid acronym confusion to the best of our ability.
- Eth1 addresses (in the form of 20-byte strings) denoted by *eth1_withdrawal_address*. As stated in [Eth22b], after the merge of the current Ethereum application layer into the Beacon Chain, withdrawals to *eth1_withdrawal_address* will simply be increases to the account's Eth balance that do not trigger any EVM execution.

We note that withdrawal credentials can be "cold": that is, they are not needed for frequent validator operations and only need to be used when moving staked Eth or staking rewards.

TIM: What is the EF using here for the CIP Nodes? I'm guessing BLS but I can't find it.

Validator Keys. The *validator keys* [Eth22b] are keys for the BLS signature scheme [BLS01] implemented with the BLS12-381 curve [Bow17]. The validator keys are used to actively sign data throughout the operation of the validator and must be "hot." These are the traditional signature keys that one associates with blockchain activity.

We emphasize that the signing key of the validator key set must be kept secure, as it cannot be changed during the operation of the vaildator (although the stake can be withdrawn and used to create a new validator).

TIM: I'm assuming that these validator keys are sampled independently from the withdrawal credentials and cannot be updated because I couldn't find text to support this. Can you confirm this? I want to make sure that I'm not missing a delegation or update mechanism, which would be awesome to have.

Transaction Fee Recipient Keys. When the validators construct blocks, they can send the transactions fees from the block (including gas fees) to a recipient that is not necessarily themselves. The *feeRecipient* is a 20-byte string that designates an Eth1 address and is a part of the *ExecutionPayload* of the Beacon Chain specification. The value of this *feeRecipient* is thus a *transaction fee recipient key*.

The execution payload structure is defined in the execution engine specification [Eth22c] as well as the Bellatrix update description [Eth22a] which is the update implementing "the merge." We note that the *feeRecipient* is equivalent to the *beneficiary* from the original Ethereum yellow paper [W+14].

References

[BG17] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BLS02] Paulo SLM Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *International conference on security in communication networks*, pages 257–267. Springer, 2002.

[Bow17] Sean Bowe. Bls12-381: New zk-snark elliptic curve construction, 2017. https://electriccoin.co/blog/new-snark-curve/.

[Con22] Consensys. Teku: The ethereum 2.0 client for institutional staking, 2022. https://consensys.net/knowledge-base/ethereum-2/teku/.

[DPS19] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *FC 2019*, LNCS, pages 23–41. Springer, Heidelberg, 2019.

[Eth22a] Ethereum. Bellatrix: The beacon chain, 2022. https://github.com/ethereum/consensus-specs/blob/dev/specs/bellatrix/beacon-chain.md.

[Eth22b] Ethereum. Ethereum consensus specification: Phase 0 – honest validator, 2022. https://github.com/ethereum/consensus-specs/blob/dev/specs/phase0/validator.md.

[Eth22c] Ethereum. Ethereum execution engine api, 2022. https://github.com/ethereum/execution-apis/blob/main/src/engine/specification.md.

[Eth22d] Ethereum. The great renaming: what happened to eth2?, 2022. https://blog.ethereum.org/2022/01/24/the-great-eth2-renaming/.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, August 2017.

[Shi20] Elaine Shi. Foundations of distributed consensus and blockchains, 2020. https://www.distributedconsensus.net.

[W+14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.