OpenST Mosaic

A FRAMEWORK TO SCALE ETHEREUM

Benjamin Bollen, Martin Schenck
for OpenST Foundation

Abstract

We present improvements to the OpenST protocol. OpenST is a framework powered by Ethereum to build token economies. We lay out in detail two contributions, OpenST Mosaic and OpenST Gateway, which work together to scale Ethereum. OpenST Mosaic is a layer-2

1. Introduction

Ethereum [1] with its current proof of work Nakamoto consensus protocol is inherently limited in the number of transactions it can perform per second [2]. In order for Ethereum to gain mass adoption, an increase in throughput is required. Current efforts include state channels [3] like the Raiden Network [4] and side chains [5] like Plasma [6]. A drawback of the proposed solutions is the fact that the user has to be always online in order to ensure integrity of her transactions. Furthermore, in case of a mass exit, a congested Ethereum network could lead to lost funds.

OpenST Mosaic is a holistic solution powered by Ethereum that we expect to securely scale Ethereum to thousands of transactions per second. We achieve that by verifiably finalising batches of sidechain transactions on Ethereum.

2. Related Work

Verifiers' Dilemma [2]

Interblockchain Communication [7]

Casper FFG [8]

3. Our Contribution

4. OpenST-Mosaic

OpenST Mosaic is not conceptually bound to Ethereum. As layer one we assume a blockchain that has byzantine fault tolerance [9] and plausible liveliness [8]. We call that blockchain *Origin* O. We chose Ethereum as Origin for our first implementation. Transactions will take place on a sidechain that we call *Auxiliary* A. Origin and Auxiliary must both

We assume Origin to track all ownership. Origin only lends ownership to Auxiliary. Thus, finality on Origin is always authoritative. If Auxiliary halts, users can recover the assets they own and there is no restriction in time for them to do so, enabling a coordinated mass exit over time.

4.1. OpenST-Mosaic

We assume without loss of generality that for every validator-address v^O in the validator set $\mathcal{V}^O$ on origin a corresponding, unique validator-address v^A on auxiliary A can be associated and we call the resulting validator set $\mathcal{V}^A$, the validator set on A. A single actor is assumed to control and be responsible for both v^O and the associated v^A. On origin v^O will always be held accountable for any signed message by v^A on A, because and only v^O has stake on origin.

Lemma 1 (Remote Accountability)**.** *A vote on A that violates a Casper Commandment can be punished on O.*

Proof. Given lemma **??**, a violation of either Casper Commandment can be proven on O without the respective transaction residing on O. If a validator $\mathbf{v}^A$ published two distinct votes for the same target height, the relevant votes can be presented on O and the signature can be validated. An offending validator $\mathbf{v}^O$ can be identified according to lemma **??** and its deposit slashed. The same is true if a validator $\mathbf{v}^A$ votes within the span of its other votes. All relevant votes including their signatures can be presented on O. □

Theorem 2 (Partial View)**.** *Given Origin O with blocks b_i^O, checkpoints c_i^O can be reported on Auxiliary A, such that a set of validators $\mathcal{V}^O$ with stake on O can reach finality about the reported checkpoints c_i^O on A as $\mathcal{V}^A$ with accountable safety enforced on O.*

Proof. If you report checkpoints from Origin as $\langle h_c, h(c)\rangle$ where h_c is the hash of any checkpoint c on Origin and $h(c)$ is the height of checkpoint c in Origin's checkpoint tree, then you can vote on Origin's checkpoints on Auxiliary according to the Casper FFG voting rules. A Casper FFG vote consists of two checkpoint hashes, two checkpoint heights, and the validator signature, all of which is now available on Auxiliary.

Accountable safety is enforced according to lemma 1. □

Theorem 3 (Leveraged Security)**.** *Given that the auxiliary chain A follows the Casper fork choice rule and given that the validators staked on Origin O, Casper can be applied to A as a whole; without validators staking on A. It is not necessary to report blocks or block headers from A to O.*

Proof. Validators staked on Origin

Lemma 1 proves that the validators can be slashed on Origin for their actions on Auxiliary. Even though Auxiliary is not necessarily lively, Casper still has plausible liveliness, as Casper assumes that the underlying chain keeps producing blocks. □

Reporting a finalised auxiliary checkpoint to Origin costs gas on Origin. Therefore, validators do not have an incentive to copy the state root from Auxiliary to Origin. We present an asynchronous mechanism that forces validators to copy the state root. We define the amount of consumed gas since the last copied checkpoint g. We define a limit of consumed gas l. The finalised auxiliary checkpoint with the smallest dynasty where $g \geq l$ must be copied to Origin.

$$\neg\exists f : h(f') < h(f) \wedge g(f') \geq l \tag{1}$$

If the validators fail to copy that finalised checkpoint, they are still incentivised to copy a later checkpoint to get rewarded (see below). When a validator copies a state root of a later finalised checkpoint from Auxiliary to Origin, then it can be proven that there exists a finalised checkpoint with a lesser height that is also above the gas limit. To do that, a fisherman could present the earlier finalised checkpoint to Origin after the later one was committed. The height, consumed gas, and validator signatures can be proven. When that happens, all validators in the validator set are punished and the fisherman is rewarded.

We must track the consumed gas on Auxiliary in order to for this proof to work.

Theorem 4 (Honesty Guard)**.** *An honest validator will not be punished by the gas limit rules.*

Proof. All validators, including the honest ones, would be punished if the lowest finalised auxiliary checkpoint above the gas limit were not reported to Origin. If there is at least one honest validator, however, that validator would report the auxiliary checkpoint to Origin. Thus, all validators get only punished if none of them is honest. □

4.2. OpenST-Gateway

OpenST Gateway enables chain-to-chain transfer of state objects.

4.3. Reward Structure

On Origin in OST.

rewarding for reported block headers that get finalised

4.4. Dynamic Set of Validators

4.5. Set-Up of An Auxiliary Chain

Start running with gas cost of 0. Set up contract with base token. Set up Gateways, but closed. Give one week of blaming on Origin. How can there be proof on Origin without Aux's state root? Or is that also transferred in the set-up phase? If so: why is it trusted? Open Gateways.

5. Outlook

5.1. Token Economies

5.2. Neo and Cardano

6. Conclusion

References

[1] Wood, G. Ethereum: A secure decentralised generalised transaction ledger (2015). URL `http://gavwood.com/paper.pdf`.

[2] Luu, L., Teutsch, J., Kulkarni, R. & Saxena, P. Demystifying incentives in the consensus computer. *IACR Cryptology ePrint Archive* **2015**, 702 (2015).

[3] Poon, J. & Dryja, T. Lightning network (2015). URL `https://lightning.network/lightning-network-paper.pdf`.

[4] Raiden network. URL `https://raiden.network/`.

[5] Back, A. *et al.* Enabling blockchain innovations with pegged sidechains (2014). URL `https://www.blockstream.com/sidechains.pdf`.

[6] Poon, J. & Buterin, V. Plasma: Scalable autonomous smart contracts (2017). URL `https://plasma.io/plasma.pdf`.

[7] Kwon, J. & Buchman, E. Cosmos: A network of distributed ledgers (2016). URL `https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md`.

[8] Buterin, V. & Griffith, V. Casper the friendly finality gadget. *CoRR* **abs/1710.09437** (2017). URL `http://arxiv.org/abs/1710.09437`. 1710.09437.

[9] Castro, M., Liskov, B. & et. al. Practical byzantine fault tolerance. In Leach, P. J. & Seltzer, M. (eds.) *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 99, 173–186 (1999).