Hello, I was brainstorming a trustless withdrawal process for pooled staking in eth2.0. Two things to note:

1. This may be useless if/when withdrawal_address can be set to a smart contract directly.
2. The security model requires a threshold set of honest pooled nodes to operate (threshold was selected over multisig for potential key loss).

Furthermore, I understand the cost of many of the required ops (BLS verify onchain) may stray from making this pragmatic in which case this is purely an academic exercise.

At the moment I am focusing solely on the withdrawal process.

[

Eth2StakedPool (1)

1301×669 72.9 KB

](https://ethresear.ch/uploads/default/original/2X/9/9f6d669e421a587234e6f0f707a1202d68055836.png)

The Idea

1. The withdrawal_address is set to a threshold BLS key where the key shards are distributed to the validator and n pool participants
2. When the withdrawal is initiated, the distributed key holder (DKH) generates a eth2.0 transaction moving funds from the shared withdrawal_address to a target_address controlled by the (DKH). The DKH signs this txData with its BLS key share.
3. The DKH submits the txData on eth1.0 HTLC smart contract and sets the value=eth2.0 withdrawal_amount
4. Since we use threshold BLS instead of multisig, m of n nodes in the pool sign txData and resubmit to HTLC. We require an honest majority to ensure that the submitted value >= the eth2.0 value
5. The HTLC contract defines the following condition:

if(txData == valid_bls_sig){

distribute funds to pool holders based on registry

}

1. The DKH receives the fully signed txData and can execute on eth2.0 to send funds to targetAddress

Issues

- we use threshold instead of multisig to prevent a withholding attack (but cause other problems)
- sybil/collusion attack - would require multiple random sub-committee selection rounds to sign txData

Not discussed and needs more thought:

- Distributed BLS KeyGen ceremony for shared withdrawal key
- Validator “bond” to prevent malicious attestations
- Trustless depositing (would likely extend from the HTLC smart contract acting as a registry and after accumulating > 32ETH calling the eth deposit contract and setting the initial withdrawal address once a set of nodes fund the contract. Of course the dkeygen between the pooled nodes needs to be performed prior to this event)