We (https://www.p0xeidon.xyz/) would love to introduce Poseidon VM to the Ethereum community. You can see the white-paper here and a brief talk here. Below is the TLDR:

Poseidon VM aims to solve two problems in the Ethereum ecosystem:

1. ZKP based applications has a high barrier to entry for builders
2. ZKP based applications has a high verifier cost (gas) as well as prover cost (while many existing L2s like OP/Arbitrum/Scoll/zkSync can scale normal Ethereum transaction, they cannot scale zk transactions very well, for example, zkopru has about 100 TPS, which is not bad at all but still limited)

Our answer to the first problem is to build layered programming abstractions

. More specifically, we make the distinction between zkApp devs and Circuit devs. They can be the same group of people but not necessarily so. Our goal is to make solidity devs can build zkApps easily without coding circuits at all! What is missing here is a “standard libraries” layer for zkApp development. And we think the VM itself should incorporate these standard libraries such as Unix provide things like cat

, less

, pipe

etc.

One critical problem need to be solved here is that these “standard libraries” need to be able composed with each other on the solidity level. However, it is not the case now for the most circuits developed today. The general framework to solve this issue is called “Commit-and-Prove SNARK” (CP-SNARK, from the LegoSnark paper). But the idea is surprisingly simple, Let’s say if you have two circuits, C1

and C2

. To let them compose with each other, essentially you want these two circuit can have some shared private state: you just commit the private state to be shared in these two circuits using the same hash function and the same trapdoor. Then, inside each circuit, the commitment is opened. You do need to make sure these two circuit have the same commitment as public input tho.

Below is an example, suppose you want to write a solidity contract for voting APEFEST 2023’s location anonymously, below is the code you can write in Poseidon VM:

```
contract ApeFest2023Voting { mapping(Proposal => uint) public votes; function anonymous_vote( bytes32 rc, Proposal proposal, bytes32 ext_null, ZKAsset.proof asset_pi, CAD.proof cad_pi, ZKSignal.proof sig_pi ) returns (bool) { // assert that the identity committed to in rc owns an asset in the ZSP require(ZSP.owns(rc, asset_pi), "invalid asset ownership"); // assert that the asset is an APE NFT require(CAD.is_erc721(rc, APE_Address, cad_pi), "asset is not an APE NFT"); // assert that the voter is the owner and // there is no double voting require(ZKSignal.verify(rc, APE_2023_EN, proposal, sig_pi), "invalid vote"); // increment vote count votes[proposal]++; } }
```

This code uses 3 “standard libraries”. First is ZSP

, zkAsset Shielded Pool, a shielded pool for ERC-20/ERC-721/ERC-1155. It allows you to prove the ownership of an UTXO in the shielded pool without revealing extra info. Second is CAD

, Configurable Asset Disclosure, which allows you to prove you own a “APE” (in this case), without disclosing which APE you exactly own. Third is ZKSignal

, which works similarly to Semaphore but without membership part (since membership part in this case is composed with NFT ownership). Now you can see, a few lines of Solidity code allows devs to build zkApps!

Our answer to the second question (scaling zkApp transactions in L2) has two part:

- building zkp friendly precompiles such as poseidon hashes, and poseidon hash based merkle trees in the storage layer
- a new L2 sequencing mechanisms to make zkApp’s L2 gas cost much lower. (we will explain this part in a separate post)

Below is the Poseidon VM’s architecture:

[

PVMArch

1655×917 27.9 KB

](https://ethresear.ch/uploads/default/original/2X/d/d83952d37d7da41f66fc89187d61a471da4a63e4.png)

Would love to hear your thought and what kind of zero-knowledge proof powered applications you want to build.