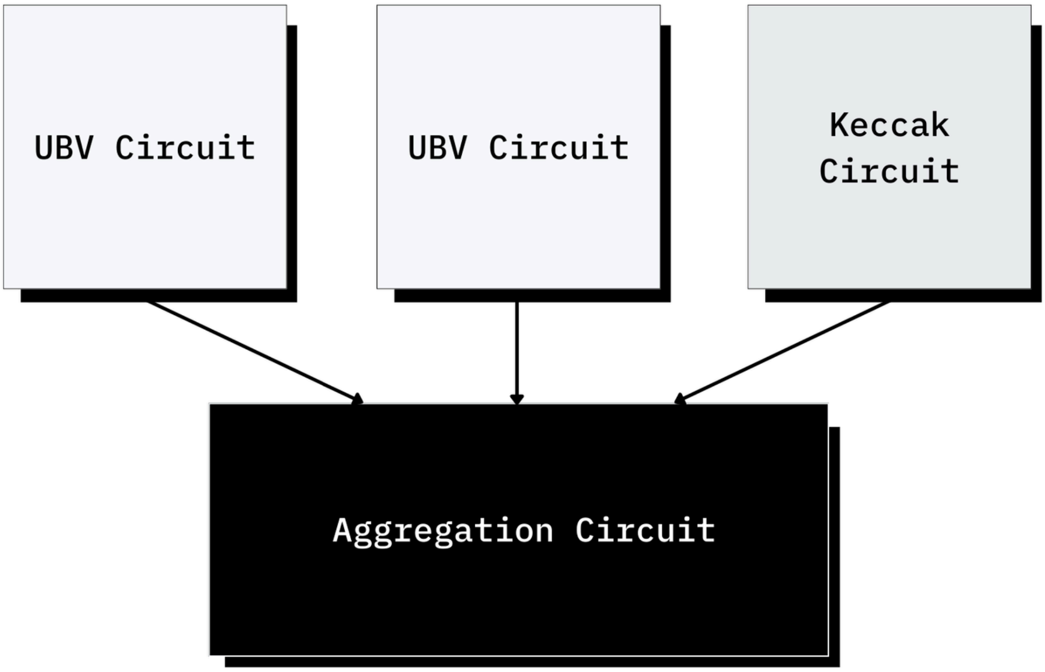
Proof Aggregation Benchmark

by NEBRA Team

NEBRA UPA v1.2

NEBRA UPA (Universal Proof Aggregation) protocol aggregates proofs from different parties into the same proof. In a typical deployment, NEBRA UPA v1.2 uses a 2 layer aggregation circuits. The first layer consists of two kind of circuits:

- **UBV *(Universal Batched Verifier)* Circuit**: Universal batched verifier circuits that stacks up *k Groth16 verifiers.*
- **Keccak Circuit**: Keccak Circuit that binds the public inputs to each proofs



The second layer consists of an Aggregation Circuit, which takes as inputs the proofs generated by the first layer, and generates an final aggregated proof that can be verified efficiently on EVM.

Methodology

We measured the performance of each zkVM running a Groth16 verifier, written in Rust using cryptographic primitives from Arkworks. The inputs are a single Groth16 proof, public inputs, and verification key over the BN254 curve. This proof is verified *N* times to simulate aggregating a batch of size *N*. The resulting ZKVM proof is required to be on-chain verifiable, when the necessary recursion circuits are available (currently only Risc Zero and SP1). All benchmark code is available here.

The UPA's performance was measured using the open-source UPA prover tool. For each batch size, we chose an appropriate circuit configuration and measured the total time required to produce an on-chain-verifiable aggregation proof from a batch of Groth16 application proofs. This implies computing proofs of the UBV, Keccak, and Aggregation circuits. All proofs from the first layer of recursion (UBV and Keccak circuits) are computed in parallel, then the second layer of recursion (Aggregation circuit) is computed. The full script for keygen and benchmarking can be found here.

To keep the benchmark simple, the zkVM-based aggregation merely verifies *N* Groth16 proofs. On the other hand, the UPA also commits to the public inputs of the input proofs so that the verification result may be accessed on-chain.

Another important difference is that the UPA supports an extension of Groth16 used by Gnark to expose commitments to private witness values. In other words, the UPA circuits implement a more complex form of aggregation than what we benchmarked for the zkVMs.

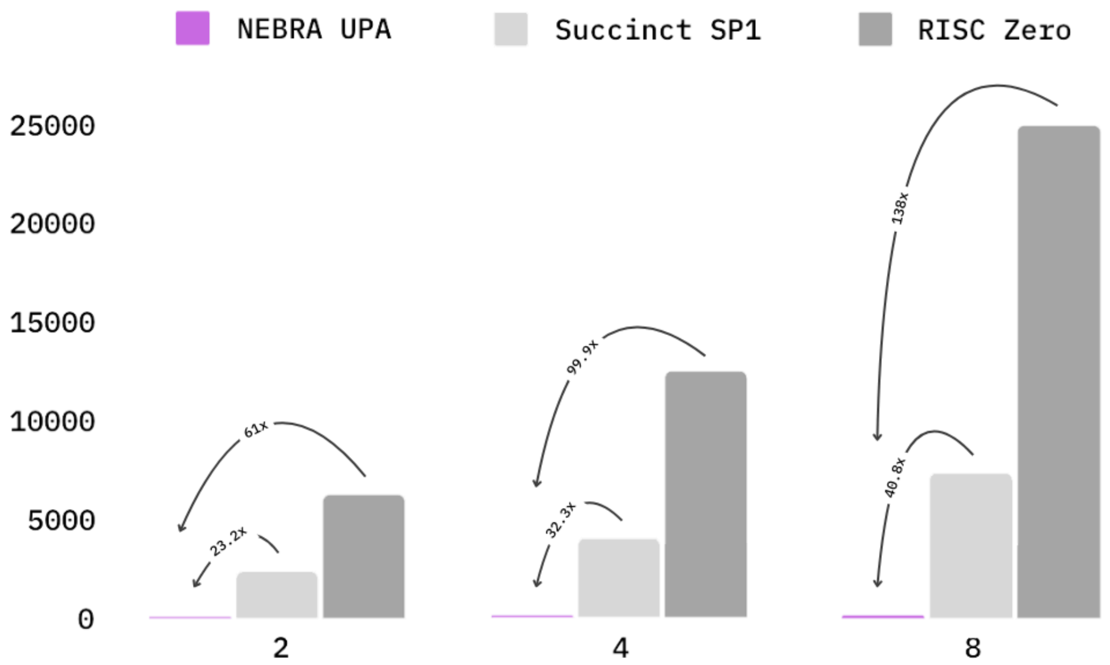
Benchmark: NEBRA UPA v.s. zkVMs

We compare NEBRA UPA with simply deploying a Groth16 verifier on major zkVMs. To make the comparison fair, ***we didn't use precompiles*** to accelerate the related cryptographic operations nor uses any GPU accelerations. For all the zkVMs, we use their latest released version. In the case that there is no released version, we uses the latest main branch version in the github repo by August 2nd, 2024.

All these experiments is performed in a dedicated server with:

- *AMD EPYC 7702 64-Core CPU*
- 1 TB RAM

Proof Aggregation Engine	2 Proof (sec)	4 Proofs (sec)	8 Proofs (sec)
NEBRA UPA	103	125	181
Succinct SP1	2,392	4,039	7,379
RISC Zero	6,283	12,483	24,984



We would like to include Jolt and Nexus in this benchmark, but have so far been unable to run our verifier program in their zkVMs. We invite them to open a PR in the benchmark repo.

Conclusion and Outlook

In this benchmark, we demonstrated that **NEBRA UPA**'s performance is **orders of magnitude** faster compared with naively performing aggregation in a zkVM. Moreover, we observe that the UPA's performance scales better with the batch size, thanks to its hand-optimized circuits and two layers of recursion.

Of course, zkVMs have other strengths, such as flexibility and ease of development. It's worth noting that although the UPA is fast, it is currently limited to Groth16 proof aggregation. In the upcoming NEBRA UPA V2, we are adopting a zkVM + precompile approach to rapidly enable support for other proof systems. Please stay tuned.