Prove keccak-256:

This includes the following steps:

1. Generate 50 stark-proofs for Keccak-f, 50 stark-proofs for Keccak-sponge, 50 stark-proofs for logic, and aggregate every tuple of 3 starks into 1, then we get 50 proofs. namely "proof_and_hash_results".
2. Generate a proof which aggregates the 50 proofs from the 1st step. namely "recursive_multiple_plonky2proofs".
3. Generate a proof which recursively compress the proof from the 2nd step to the minimal size. namely "compress".
4. Generate a proof with bls12377-poseidon_hash config which recursively prove the proof from the 3rd step, this step only aims to align with the bls12-377 in varuna. namely "recursive_bls12377plonky2_proof".
5. Wrap the proof from the 4th step into Varuna.

The time taken for 1-4 steps can be obtained by checking the output of programs for fields similar to "prove_and_aggregate" and the 5th step by "Varuna::Prover".

Here is an example:

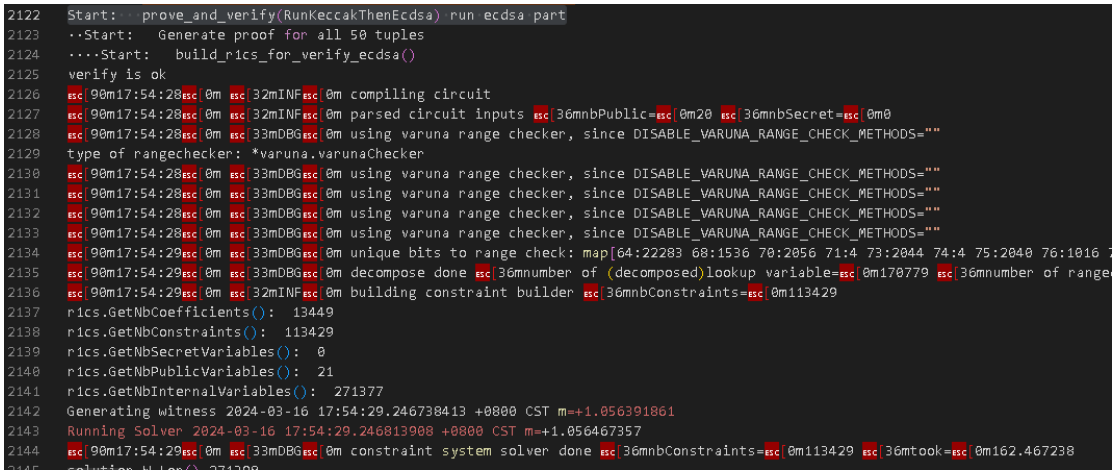
```
1018  838.8387s to prove_and_aggregate
1019  | 676.1342s to proof_and_hash_results
1020  | 38.6164s to recursive_multiple_plonky2proofs
1021  | | 6.7663s to Building circuit time:
1022  | | 10.7449s to generate recursion 1 plonky2 proof for 50 starks
1023  | | 6.9090s to Building circuit time:
1024  | | 10.5843s to generate recursion 1 plonky2 proof for 50 starks
1025  | | 0.6999s to Building circuit time:
1026  | | 1.1634s to generate recursion 1 plonky2 proof for 50 starks
1027  | 29.2671s to compress
1028  | | 4.3646s to Building circuit
1029  | | 15.0882s to generate single plonky2 recursive proof to compress size
1030  | | 2.0101s to Building circuit
1031  | | 7.6595s to generate single plonky2 recursive proof to compress size
1032  | 89.6540s to final recursive_bls12377plonky2_proof
1033  | | 18.9906s to Building circuit
1034  | | 70.6152s to final recursive_bls12377plonky2_proof
1035  ··End:     plonky2_evm::hash2::prove_and_aggregate .....838.915s
```

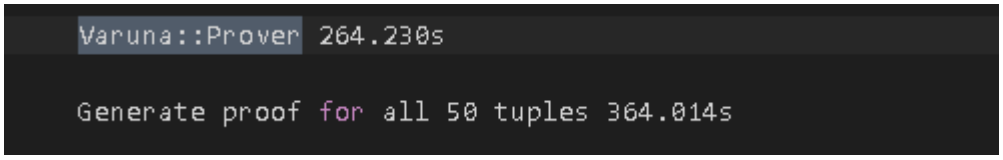
```
1991  ·····End:     Opening 11 polynomials at query set of size 10 .....20.059s
1992  ····End:     Varuna::Prover .....115.396s
1993  ··End:     Generate proof for all 50 tuples .....124.732s
1994  ··Start:   Verify proof for all 50 tuples
```

Prove ECDSA:

Time taken for ECDSA proof generation can be obtained by checking the output of programs for fields similar to “Varuna::Prover” after this line:
”start: prove_and_verify(RunKeccakThenEcdsa) run ecdsa part”.

Here is an example:





This is the time to generate 50 ECSDA verification proofs.