

## 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_ewm::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:

```

2122 Start: prove_and_verify(RunKeccakThenEcdsa) run_ecdsa part
2123 --Start: Generate proof for all 50 tuples
2124 ....Start: build_rics_for_verify_ecdsa()
2125 verify is ok
2126 [asc 90m17:54:28][asc 0m [asc 32m]INF[asc 0m compiling circuit
2127 [asc 90m17:54:28][asc 0m [asc 32m]INF[asc 0m parsed circuit inputs [asc 36m]mbPublic=[asc 0m20 [asc 36m]mbSecret=[asc 0m0
2128 [asc 90m17:54:28][asc 0m [asc 33m]DBG[asc 0m using varuna range checker, since DISABLE_VARUNA_RANGE_CHECK_METHODS=""
2129 type of rangechecker: *varuna.varunaChecker
2130 [asc 90m17:54:28][asc 0m [asc 33m]DBG[asc 0m using varuna range checker, since DISABLE_VARUNA_RANGE_CHECK_METHODS=""
2131 [asc 90m17:54:28][asc 0m [asc 33m]DBG[asc 0m using varuna range checker, since DISABLE_VARUNA_RANGE_CHECK_METHODS=""
2132 [asc 90m17:54:28][asc 0m [asc 33m]DBG[asc 0m using varuna range checker, since DISABLE_VARUNA_RANGE_CHECK_METHODS=""
2133 [asc 90m17:54:28][asc 0m [asc 33m]DBG[asc 0m using varuna range checker, since DISABLE_VARUNA_RANGE_CHECK_METHODS=""
2134 [asc 90m17:54:29][asc 0m [asc 33m]DBG[asc 0m unique bits to range check: map[64:22283 68:1536 70:2056 71:4 73:2044 74:4 75:2040 76:1016 7
2135 [asc 90m17:54:29][asc 0m [asc 33m]DBG[asc 0m decompose done [asc 36m]number of (decomposed)lookup variable=[asc 0m170779 [asc 36m]number of range
2136 [asc 90m17:54:29][asc 0m [asc 32m]INF[asc 0m building constraint builder [asc 36m]mbConstraints=[asc 0m113429
2137 rics.GetNbCoefficients(): 13449
2138 rics.GetNbConstraints(): 113429
2139 rics.GetNbSecretVariables(): 0
2140 rics.GetNbPublicVariables(): 21
2141 rics.GetNbInternalVariables(): 271377
2142 Generating witness 2024-03-16 17:54:29.246738413+0800 CST m=+1.056391861
2143 Running Solver 2024-03-16 17:54:29.246813908+0800 CST m=+1.056467357
2144 [asc 90m17:54:29][asc 0m [asc 33m]DBG[asc 0m constraint system solver done [asc 36m]mbConstraints=[asc 0m113429 [asc 36m]mtok=[asc 0m162.467238
2145 solver.H.Loss() 0.74309

```

```
Varuna::Prover 264.230s  
  
Generate proof for all 50 tuples 364.014s
```

This is the time to generate 50 ECSDA verification proofs.