I wanted to get some sense of how fast the new hash functions (Rescue and Poseidon) are - so, I put together a fairly performant implementation of both in Rust (here). As far as I know, this is the fastest implementation of these hash functions - but if anyone knows any faster implementations, let me know.

results

These results are from for Intel Core i5-7300U @ 2.60GHz (single thread):

- Rescue: ~12,000 hashes/second or ~80,000 ns / hash
- Poseidon: ~33,000 hashes/second or ~30,000 ns / hash

For comparison, on the same machine, I can do over 1M sha256 hashes/sec. So, Poseidon is almost 3x faster than Rescue, and both are significantly slower than sha256 (this was expected). Specifically, Poseidon is about 30x slower, and Rescue is about 80x slower than sha256.

parameters

The parameters I used for hash functions are similar to the ones listed for S128d variant in StarkWare's hash challenge. Specifically:

- Rescue: 10 rounds with state width of 12 64-bit field elements
- Poseidon: 48 rounds (8 full / 40 partial) with sate width of 12 64-bit field elements

The main difference between the parameters I used and the ones listed on the hash challenge site is the field. I'm using a 64-bit prime field, while StarkWare challenge uses a 62-bit prime field (64-bit field made things a bit faster for me).

The use of a 64-bit field probably means that this implementation cannot be used inside pairing-based SNARKs, but it should still be usable in STARKs.

performance analysis

Performance of these functions can be improved further. Here is what's taking the most time in each hash function:

- Rescue: 85% of the time is spent doing the inverse S-Box, and 14% of the time is spend on applying MDS matrix.
- Poseidon: 92% of the time in spent on applying MDS matrix, and 5% of time is spent in doing the S-Box.

In my implementation, exponentiation is not particularly optimized - so, optimizing it should improve performance of Rescue. For Poseidon, the improvements could come from reducing the number of rounds. This could potentially be done by increasing S-Box degree from 3 to 5 - though, I'm not sure how many rounds this would allow to shave off.

Edit:

Additional results:

- GMiMC_erf: ~380,000 hashes/second or ~2,600 ns / hash

This makes GMiMC hash function by far the fastest one out of the 3. It is almost 12x faster than Poseidon, and only about 2.5x slower than sha256.