I've just released a new version of Distaff VM. In this update I focused on adding many useful instructions - so, now Distaff is almost a fully-functional VM.

The current instruction set is described here in detail, but here are the highlights:

1. Added hashing operation. This operation computes a single round of Rescue hash function. Executing this operation 10 times in a row is equivalent to computing a full Rescue hash (at ~120-bit security level). You can hash up to four 128-bit elements at a time (or about 512 bits total).

2. Implemented support for secret inputs. A program can now consume an unlimited number of secret inputs. This is done via two input tapes (these are creatively named tape A

and tape B

). You can push values from both tapes onto the stack with a single instruction. Since each value is an element in a 128-bit field, you can move up to 256 bits onto the stack in a single operation.

1. Added a bunch of stack manipulation operations and a few conditional selection operations. These make it possible to write simple conditional logic. Also, completed the set of arithmetic operations. Specifically, added INV

instruction to compute multiplicative inverses (so, now you can do divisions).

Using these new instructions you can write many useful programs. Specifically, writing a program which can anonymously prove membership in a Merkle tree is pretty simple (I have an example of such a program here). This means that Distaff VM can be now be used to support Semaphore-like protocols (as well as many other things).

Impact on performance

As expected, adding new instructions to the VM slowed down proving time. But the impact was not as significant as I expected. Overall, proving time increased by about 25%. However, the performance is still pretty impressive. For example, the VM can generate proofs for computing over 200 Rescue hashes / sec (on a single core).

Below are some rough benchmarks for running a program which verifies Merkle proofs for trees of various depths (run on Intel Core i5-7300U @ 2.60GHz, single thread):

Tree depth

Operation count

Execution Time

Verification time

Proof size

16

29

180 ms

2 ms

73 KB

32

210

300 ms

2 ms

85 KB

64

211

600 ms

3 ms

95 K B

These proofs were done at 120-bit security level and optimized for proving time. But, if you are willing to accept 100-bit security level and spend a bit more time on proof generation, proof sizes can be reduced quite a bit. For example, for a tree of depth 32, if we increase proving time to ~1 second, the proof size can be reduced to under 50 KB.

What is missing

There are a few important things that are still missing from Distaff VM:

1. Equality and comparison instructions (less than, greater than etc.). I have a pretty good idea of how to implement these. Equality is easy and can be done in a single operation. Comparison is pretty easy as well, although it will require many operations. So, value comparisons will be one of the most expensive operations on the VM.

2. Random access memory. I have a pretty good idea of how to do this as well - though it will probably slow down the rest of the VM by 20% - 25%.

3. Ability to produce large number of outputs (currently, number of outputs is limited to 8). Not sure about the bet way to do this yet.

If you can think of anything else that could be added to the VM to make it more useful - let me know. I'm always looking for feedback and suggestions.