We can define a version of MIMC that works as follows: SimplifiedMiMCHash(x, d) = f^{512}(x)

, where f(x) = x^3 + d

; that is, we apply the permutation x \rightarrow x^3 + d

512 times.

Security claim: partial collision resistance - if y = SimplifiedMimChash(....SimplifiedMiMCHash(SimplifiedMiMCHash(x, d1), d2)...dn)

it is infeasible to find (d1', d2' ... dn') \ne (d1, d2 .. dn)

such that y = SimplifiedMimChash(....SimplifiedMiMCHash(SimplifiedMiMCHash(x, d1'), d2')...dn')

[NOTE: I think there are better ways to do this that more directly lean on traditional collision resistance properties of these arithmetically cheap hash functions…]

We now define the accumulator as follows. The accumulator A

starts at 0, and then every time a value v

is added we set A := SimplifiedMiMCHash(A, v)

.

For proofs of inclusion or exclusion, we set up a STARK with three tapes: the accumulator state A

, the witness W

consisting of a sequence of 512-value repeats of values that get added to the accumulator, a loop progress counter M

which starts at 1 and a product trace P

which starts at 1. Let \omega

be a 512th root of unity, and x

be the value you want to prove inclusion or exclusion of. We add the following constraints:

- M[i] = 1

or W[i] = W[i-1]

(ie. W

is only allowed to change at multiples of 512)

- M[i] = M[i-1] * \omega

(incrementing M

; note that it loops around to 1 every 512 steps)

- A[i] = A[i-1]^3 + W[i]
- P[i] = P[i-1] * (x - W[i-1])

We check the boundary conditions (i) A[0]

is the starting accumulator, (ii) A[n]

is the ending accumulator, (iii) P[0] = 1

. The goal is that P

will stay nonzero as long as x

is never used in the witness, and will permanently become zero if x

is used in the witness even once.

The STARK construction is very simple, with only 4 state objects to worry about; it should not be difficult to convert the existing MIMC-STARK code to implement this construction. Note that it should be fairly straightforward to replace MIMC in

this construction with Jarvis. This means that we can use STARKs for proving history inside of Plasma, and even potentially to prove contract non-double-resurrection for sharding, more quickly than a fully complete STARK system that supports more complicated operations.