Is this scheme secure?

Let H(x)

be a hash function with the following property:

H(x+y) = H(x) + H(y)

while being collision and preimage resistant.

Define the accumulated hash of a set inputs

as:

setHash = H(sha3(input_1)+sha3(input_2) +\ ...)

equivalently

setHash = H(sha3(input_1))+H(sha3(input_2)) +\ ...

then the proof of existence of an element a \in inputs

is a pair: (a, restSum)

that satisfies:

setHash = H(sha3(a)) + H(restSum)

Batch proofs are possible with one shared restSum

.

The lost functionality compared to merkle trees is enumeration.

Security:

if a \notin inputs

but proof for a

's existence is valid, it means that either:

1. collision resistance of sha3 is broken, it’s feasible to find such a, a'

that:

sha3(a) = sha3(a')

for a \ne a'

and a' \in inputs

1. collision resistance of a composite function H\ .\ sha3

is broken:

H(sha3(a)) = H(sha3(a'))

for a \ne a'

and a' \in inputs

1. preimage resistance of H is broken, it’s feasible to find invalidRestSum

such that:

setHash = H(sha3(a)) + H(invalidRestSum)

1. it’s feasible to generate {b_1, b_2, ..., b_n} \notin inputs

, n \geq 2

, d \in inputs

such that:

$\sum_{i=1}^{n} sha3(b_i) = sha3(d)$

which breaks sha3's indifferentiability from random oracle assumption

Is there an attack that doesn't require breaking hash functions?