Vitalik recently tweeted this problem

Blockquote

Cryptography problem: make an invertible, easy-in-both-directions function from [0…2^256 - 1] to [0…2^256 - 2] (or more generally, -k for small k) where the exceptions to the invertibility are computationally infeasible to find.

Vitalik's motivating use cases

Blockquote

A really efficient solution could actually significantly simplify Verkle trees, erasure codes and many other constructions in ethereum, by making 32-byte chunks one-to-one mappable to F_p elements.

The author attempts a solution with efficient verification, but one that requires preprocessing, and therefore is only as infeasibly hard to crack as the preprocessing How to hide pigeonhole collisions in 1 tweet

Expanded Solution:

Blockquote

Let function f1(x)=S1(x) yield a set of pigeon hole placements like f1(x) for any pigeon x

Let x be an integer, an index, into the range of interest, e.g. [0…2^256]

Let S1(x)=the sum of prime factors P1(x) for large integer L1(x)

S1 is a set of such sums

L1 is a set of such large integers

Let function f2(x)=S2(x)-k for other large integer L2(x) in set L2

k = the number of extra "pigeons", which is 1 in Vitalik's example

Let integers set D=S2-S1, so that D(x)=S2(x)-S1(x) for P2(x) L2(x) and P1(x) L1(x)

D is the a set of integers to shift pigeon positions from f1 to f2 and vice versa

C is a set of integers we add to D to resolve unwanted collisions into unused holes for any pigeon x

Let F(D,C,f1,f2,x) be the easily bi-invertible function from f1 to f2 and vice versa

Assign very large integers to sets L1 L2 at random

Let boolean b = (f1==f2+D-C-k) = collision if b is true

For easy verification of b, we can simply give away the prime factor keys

We sum

up the prime factors in a large number set L to get sums S for pigeon hole placement set f.

As extra unwanted collisions can occur, we can resolve these into unused holes and keep track of this in C(x) for each D(x)

Keeping prime P's a secret makes it hard for attackers to verify S and delta D.

We can make verifying S & D easy by just giving these P prime keys away for sets P1 P2. This reduces the complexity to mainly just multiplying primes in P to verify that they give L with S and D. C(x) can be checked for any collision offset to add to delta D(x).

A more verbose blog is linked here

-Gary Bernstein