#ret #incomplete #hw *****

1 | Cryptography!

1.1 | Hashes

1.1.1 | Requirements for a hash

First, how do we know if it works? A hash needs to be: - One way? - Deterministic - Unique

How we we prove it is one way? uh, we can't. unless we prove P!=NP. hash function zoo! `https://ehash.iaik.tugraz.at/wiki/The_Hash_Function_Zoo`

1.1.2 | -

source - No preimage: given *y*, it should not be feasible to find *x* such that *h(x) = y*. - No second preimage: given *x1*, it should not be feasible to find *x2* (distinct from *x1*) such that *h(x1) = h(x2)*. - No collision: it should not be feasible to find any *x1* and *x2* (distinct from each other) such that *h(x1) = h(x2)*.

- what this means
  - not feasible to get the original from the function output
  - not feasible to find a colliding hash?
  - not feasible to find collisions
- breaking a hash function, from here

```
title: what does it mean for a hash function to be broken?

"For a hash function with a _n_-bit output, there are generic attacks (which work regardless of the deta
```

1.2 | Custom hashing function

what if… we just use a neural network?

create a giant, randomly initialized neural network. then, have permuting layers in the middle which make the output space non-continuous