Formal models - Distinguishing games

Sometimes cryptographers do thought experiments as a way of arguing that their cryptography is strong. When the thought experiment is to imagine you give an adversary one of two objects and the adversary has to guess which object they've been given, then you have set up a "distinguishing game".

- Flip a coin and give adversary A an object of Type 1 or 2
- Adversary A interacts with object
- Adversary A guesses 1 or 2

The adversary's advantage in this game is defined as

```
 Advantage = Pr[A guesses 1 | obj is Type 1] - Pr[A guesses 1 | obj is Type 2]
```

Notes:
- This is Pr[A right] - Pr[A wrong]
- The first part of each probability must match (ie, "A guesses 1" in both)
- A randomly guessing 1 or 2 yields Advantage 0 and A always right yields 1.
- Advantage 0 indicates the adversary does not distinguish between the two objects

Ex 1

A device has either 2 six-sided dice or 1 twelve-sided die and can be activated only once. Upon activation the device reports the result of a random roll as a single number. Give an algorithm that guesses "2x6" or "1x12". What is the algorithm's advantage?

We know that 2x6 cannot generate 1, so a simple algorithm does slightly better than just guessing is

```
    x = activate device
    if (x == 1)
      return "1x12"
    else
      return "2x6"
```

What is the advantage of this distinguishing algorithm?

```
 Advantage = Pr[Guesses 1x12 | Is 1x12] - Pr[Guesses 1x12 | Is 2x6]
           = 1/12 - 0
           = 1/12
```

The adversary could do better by guessing "1x12" on events that have a higher probability when using a 1x12 than when using 2x6. For example, when a 2x6 is in use, the outcomes 1, 2, 3, 11 and 12 have probabilities of 0, 1/36, 2/36, 2/36 and 1/36 (respectively), but when a 1x12 is in use all these probabilities are 1/12 which is greater. So, on 1, 2, 3, 11 and 12 there is a slight inference that a 1x12 is in use. So, a slightly modified distinguishing algorithm would be

```
    x = activate device
    if (x == 1, 2, 3, 11, or 12)
      return "1x12"
    else
      return "2x6"
```

in which case the advantage would be

```
 Advantage = Pr[Guesses 1x12 | Is 1x12] - Pr[Guesses 1x12 | Is 2x6]
           = 5/12 - (0 + 1/36 + 2/36 + 2/36 + 1/36)
           = 15/36 - 6/36
```

```
           = 9/36 = 1/4

Ex 2
----

A device is, with equal likelihood, loaded with either a random function from
unsigned int to unsigned int or a random permutation from unsigned int to
unsigned int. (Note that you can think of unsigned int either as strings of 32
bits or as integers in the range 0..2^32-1.) You are allowed to feed the box q
inputs and you get back q outputs.

To design a distinguishing algorithm, try to find events that are more likely in
one world than the other.

If q=1 the output is uniformly distributed in both worlds, so no event is more
likely in one world than the other.

If q=2

  x1 = device(1)   // x1 is uniformly distributed in both worlds
  x2 = device(2)   // x2 can't be x1 if device is a permutation
  if (x1==x2)
     return "function"
  else
     return "permutation"

Adv = Pr[guess function | is function] = Pr[guess function | is permutation]
    = 1/2^32 - 0

For arbitrary q:

  xi = device(i)  for i = 1..q
  if (xi==xj) for any i!=j
     return "function"
  else
     return "permutation"

Adv = Pr[guess function | is function] = Pr[guess function | is permutation]
    \approx q^2/2^32 - 0

So, a random permutation is hard to distinguish from a random function so long
as the domain is large and the number of uses is not very big.
```