The birthday problem and its application to breaking cryptographic hash functions.

What is a hash function?

A hash function is a function f that takes an arbitrary-length input m, and produces a fixed-length output. The output f(m) is known as a hash value (referred to as simply a hash).

$$F : \{0,1\}^{\infty} \rightarrow \{0,1\}^{n}.$$

A small change in m resuls in a large change in f(m)

"hello world" → 5eb63bbbe01eeed093cb22bb8f5acdc3
"hello worle" → 18c5650581f01f1a52c87eee5baa754a

(md5 is used in the example)

In a perfect system, the output of the hash function is uniformly distributed over all inputs.

Breaking hash functions.

So how can we break hash functions? Well there are three main ways off attacking a hash function

Pre-image attack
Given the hash value h, recover any m such that h = hash(m).

Second pre-image attack:
Given an input m_1, find another input m_2 (such that $m_1 \neq m_2$) such that $hash(m_1) = hash(m_2)$.

Collision attack
Find two different messages m_1, m_2 such that $hash(m_1) = hash(m_2)$.

The birthday problem.

Suppose we have a room of m people what is the probability that two people will share a birthday? This problem can be more generally described using an urn. So suppose we have an urn with balls numbered 1 to m. If we take out the balls and note down the number, what is the probability of drawing the same ball twice. We call this a collision.

The probability that the *ith* ball is different from all the others is $1 - \frac{i}{m}$.

$$\text{P(no collisions)} = \prod_{i=o}^{k-1} (1 - \frac{i}{m})$$

if $k << m : 1 - \frac{i}{m} \approx e^{-\frac{i}{m}}$ (using the taylor expansion of e^{-x})

$$\text{P(no collisions)} \approx \prod_{i=o}^{k-1} (e^{-\frac{i}{m}})$$
$$= e^{-\sum_{i=0}^{k-1} \frac{i}{m}}$$
$$= e^{\frac{-k(k-1)}{2m}}$$
$$\approx e^{-\frac{k^2}{2m}}$$

$$\text{P(collision)} = 1 - \text{P(collision)}$$
$$= 1 - e^{-\frac{k^2}{2m}}$$

let $p = P(collision)$

$$\ln(1-p) = e^{\frac{-k^2}{2m}}$$
$$2m\ln(1-p) = -k^2$$
$$k = \sqrt{-2m\ln(1-p)}$$

when p = 0.5, m = 365 (as in the original birthday problem), $\sqrt{-2 * 365 * \ln(1 - 0.5)} = 22.49$ (2 d.p).

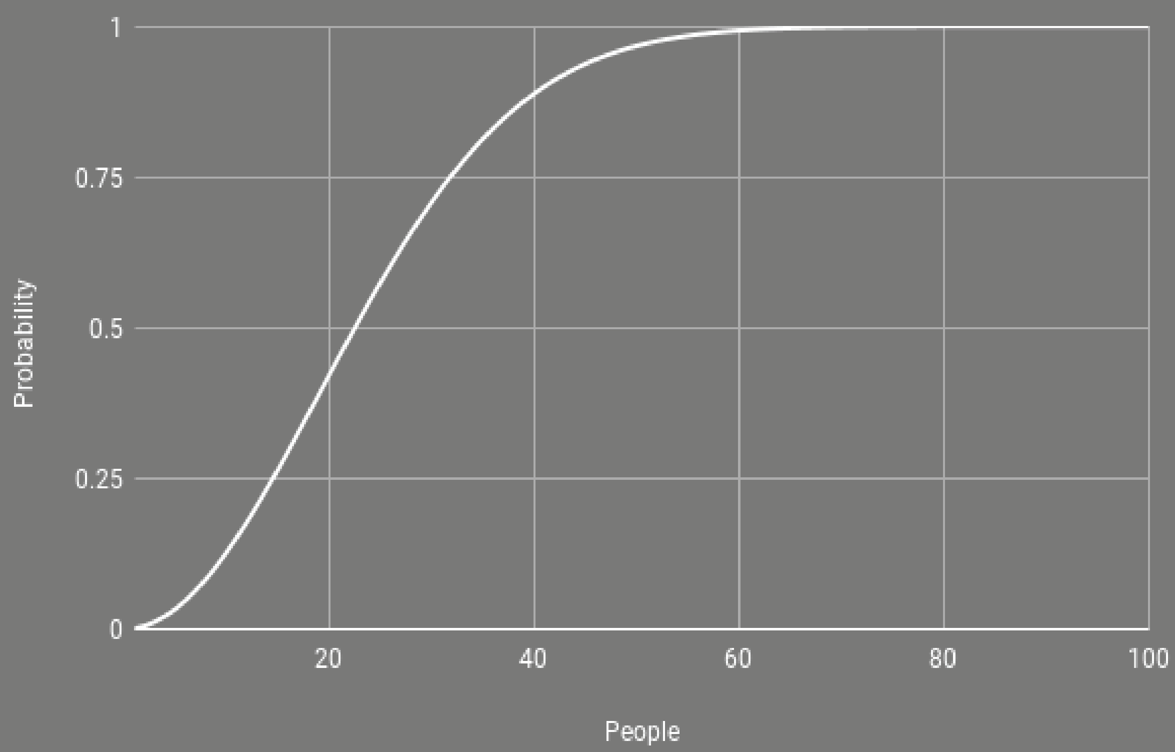
Generating collisions using the birthday problem.

We can perform a collision attack using the birthday party principle. If we say that the birthday bounds is the probability of the random collision. This means that on average we have a much faster brute force system than we did before. As we can see from the table below, the search space of our algorihtm is a lot smaller than the traditional brute force system.

Birthday bound table

Probability of random collision (p)

Bits	Possible outputs(m)	0.001	0.01	0.25	0.5	0.75
16	2^{16}	11	36	190	300	430
32	2^{32}	2900	9300	50000	77000	110000
64	2^{64}	190000000	610000000	3300000000	5100000000	7200000000
128	2^{128}	$8.3E+17$	$2.6E+18$	$1.4E+19$	$2.2E+19$	$3.1E+19$
256	2^{256}	$1.5E+37$	$4.8E+37$	$2.6E+38$	$4E+38$	$5.7E+38$
512	2^{512}	$5.2E+75$	$1.6E+76$	$8.8E+76$	$1.4E+77$	$1.9E+77$



References:
http://math.sun.ac.za/wp-content/uploads/2011/10/mmile_writeup.pdf
http://www.winlab.rutgers.edu/comnet2/Reading/documents/Birthday_attack.pdf
http://wdsinet.org/Annual_Meetings/2017_Proceedings/CR%20PDF/cr88.pdf
http://www.pumj.org/docs/Issue1/Article_3.pdf
https://blockgeeks.com/guides/cryptographic-hash-functions/

David Jay