Hash Functions

**Idea:** $H : \{0, 1\}^* \rightarrow \{0, 1\}^n$

Strings of any (finite) length  
Strings of some fixed length $n$

$n \in \{128, 256\}$

Instead of dealing with long strings, deal with hash of strings

**Ex.** MAC scheme that supports msgs of $n$ bits extend it to strings of arbitrary length via $\text{MAC}(k, H(m))$

$\Rightarrow$ If $(m^*, t^*)$ is a forgery $\Leftrightarrow t^* = \text{MAC}(k, H(m^*))$

Case 1: previously saw $m$ s.t. $H(m) = H(m^*)$  
(shouldn't happen by hash property)

Case 2: never called MAC on $H(m^*)$ $\Rightarrow$ forgery of underlying MAC ($n$-bit strings)  
(shouldn't happen by MAC property)

**Def:** $x \neq y$ are a collision under $H$ if $x \neq y$ and $H(x) = H(y)$
Obs: collisions exist unconditionally but we just need them to be "hard to find"

3 flavors

1. Target collision-resistance:
   - given $H(x)$, find $x'$ (possibly $x' = x$) such that $H(x') = H(x)$

2. Second-preimage resistance
   - given $H(x)$ and $x$, find $x' \neq x$ such that $H(x') = H(x)$

3. Collision-resistance:
   - find any collision

How hard is it to brute-force each of these games?

hardest possible $H$ to break would be one that's chosen uniformly from set of all $\mathbb{E}_0, \mathbb{E}^* \to \{0,1\}^m$

why? for each input $x$, $H(x)$ chosen indep. of everything else
   $\Rightarrow$ no structure to exploit
   (can't expect real-world hash func to be harder than this)

1. Target C.R.
   - each time you evaluate $H(x')$ for a new $x'$, have $\frac{1}{2^n}$ chance of "hitting" $H(x)$ $\Rightarrow$ try expected $2^n$ times
2. Second-preimage

Same as above,
\( \frac{1}{2^n} \) prob. for each \( x \) tried
\[ \Rightarrow \ 2^n \text{ expected trials} \]

3. Collision-resistance ("birthday problem")

**Fact:** In room of 23 people, have 50% chance of 2 sharing a birthday

**Note:** "prob that 2 people have birthday June 19?"
"prob that 2 people have same birthday?"

**Idea:** With \( N = 2^n \) possible outputs, need about \( \sqrt{N} \) trials to find collision
\[ \approx 2^{n/2} \]