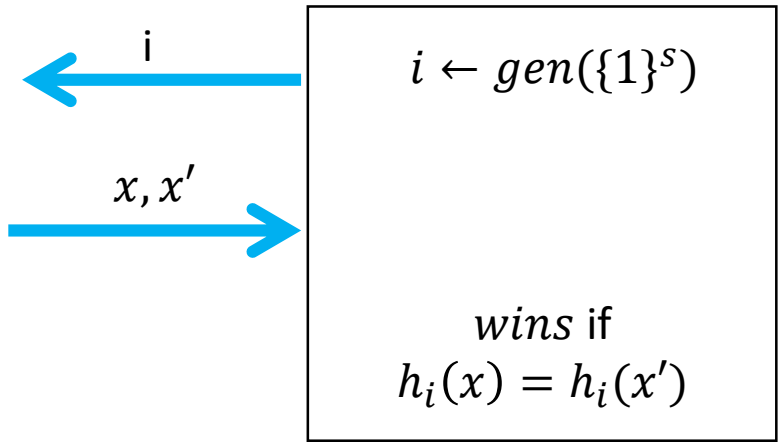
Cryptographic hash functions

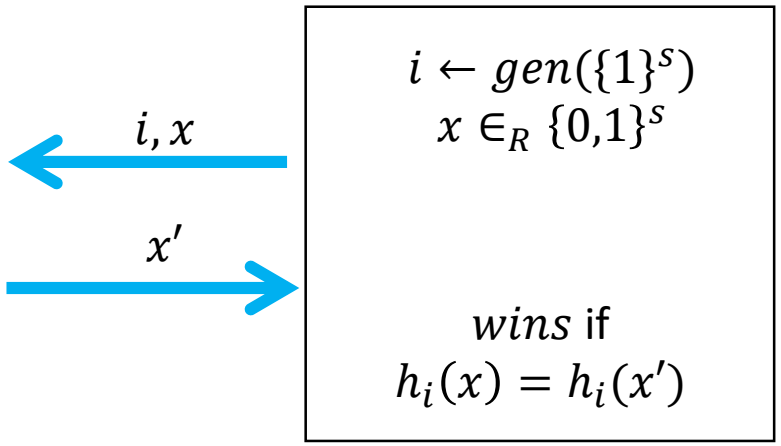
Definition of hash function

- $H := \{h_1, \dots, h_m\}$
  - $h_i : \{0,1\}^n \to \{0,1\}^{\ell(n)}$
- Compression *:* $\ell(n) < n$
- $Gen(1^s)$ picks an index between 1 and m.

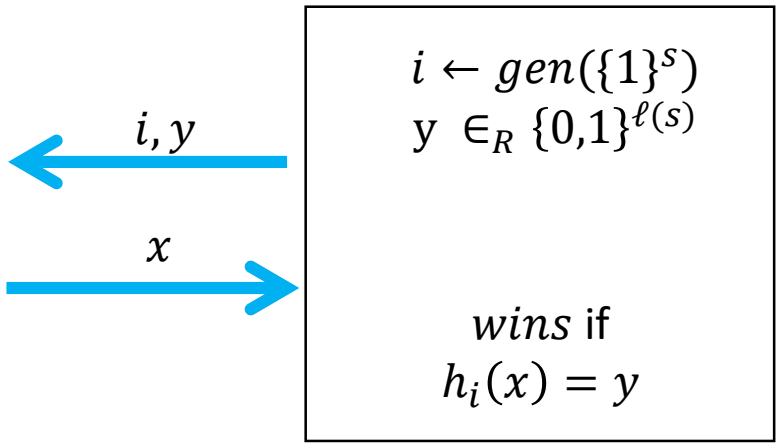
Collision resistance



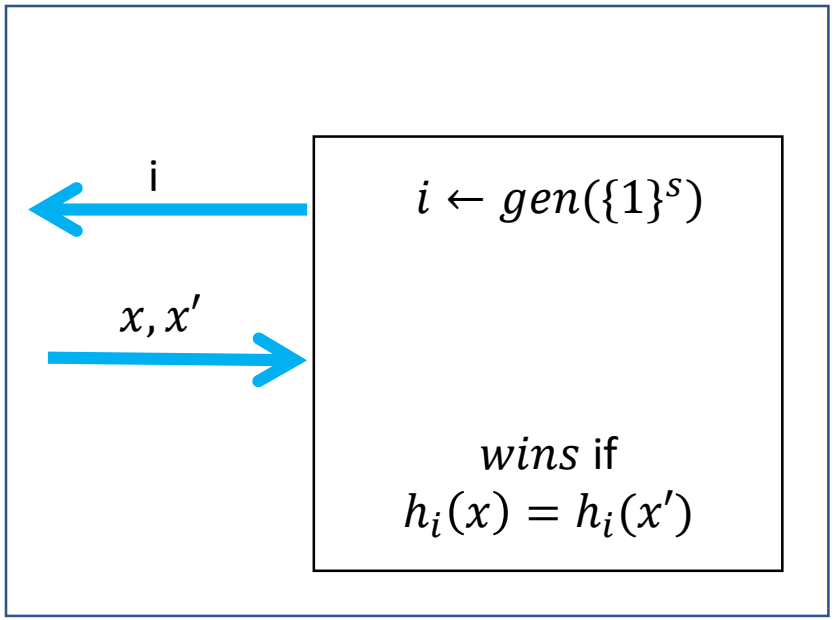
Second-collision resistant



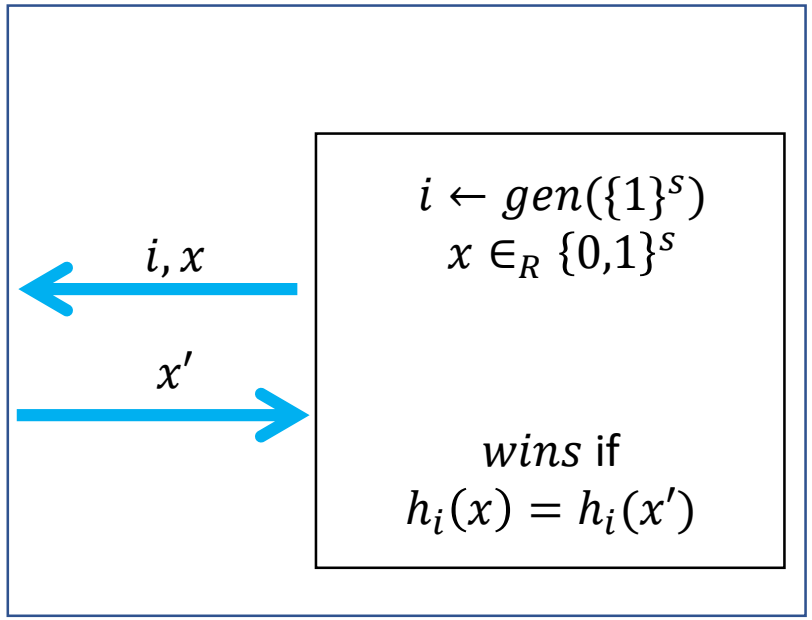
Preimage resistance



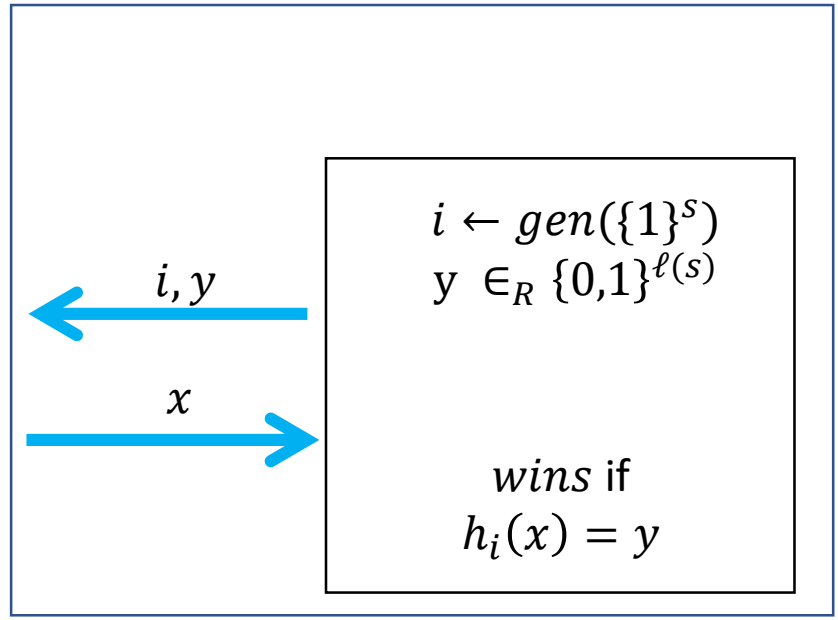
Comparison between different security



$collision\ resistant$



$target - collision\ resistance$



$preimage\ resistant$

Birthday attack

- What is the probability that in a class of 23 people there are at least two people with the same birthday.
  - Answer: more than 50% chance

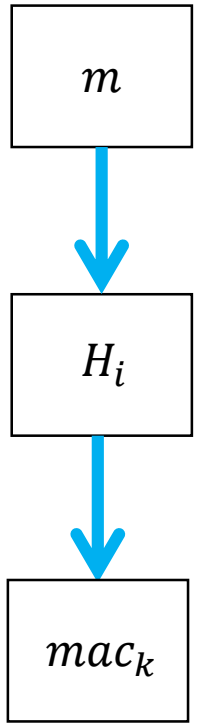
- General question: Suppose you sample m values from n values, what is the probability that there exists at least two values that are the same.
  - Answer: if $m = \sqrt{n}$ then probability is about one-half

Proof of the birthday attack

- If you store $\sqrt{n}$ output with a given input the probability that a new input gets sent to a previously visited output is $1/\sqrt{n}$
  - The probability that $\sqrt{n}$ elements all get mapped to fresh output is

$$\left(1 - \frac{1}{\sqrt{n}}\right)^{\sqrt{n}} \geq 1/e$$

Mac using hash functions



HMAC

- Global constants v_1, v_2
- Gen
  - $s \leftarrow Gen(1^s)$
  - $k \in_R \{0,1\}^s$
- Auth(m)
  - $w_1 \leftarrow (k \oplus v_2, m)$
  - $w_2 \leftarrow H^s(w_1)$
  - $w_3 \leftarrow (k \oplus v_1, w_1)$
  - $t \leftarrow H(w_3)$

Random oracle heuristic

- Assume that a hash function acts as a random function
- Allows us to prove security for efficient schemes
- Unsound but
  - Only for contrived example
  - Never broken for practical schemes

Applications of hash functions

- Virus fingerprinting
- Deduplication
- Password hashing
- File changes/integrity

Virus fingerprinting

- Hash the virus using the hash function
- To lookup a virus, simply look at the output of the function and see if it maps to a known virus
- False positives imply collision

Deduplication

- Avoid storing the same thing in memory many times

- Uses hash function to index values so that we don’t need to copy the same thing many times.

File changes / integrity

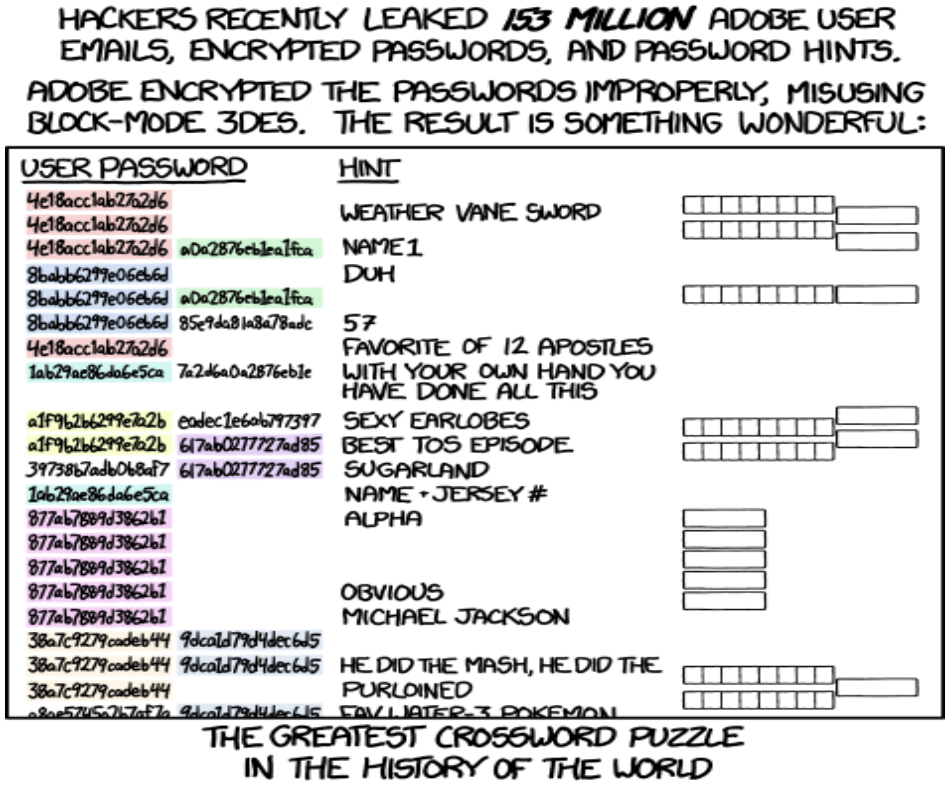
- To keep track of changes, we keep a list of hash for every function
- The output of the hash function can be much shorter than the size of the files

Proper way to hash passwords

- Naïve way to hash passwords
  - $h \leftarrow H(pwd)$
  - Same password hashed to same value

- Correct way to hash a password
  - $r \in_R \{0,1\}^n$
  - $h \leftarrow H(r, pwd)$
  - (r, h)

Bad way to hash passwords (xkcd)



Hash tree

- Goal
  - Client has constant overhead
  - Server holds client's database $x_1, \ldots, x_n$
  - Client requests x_i
  - Server sends x'_i
  - How can client verify that $x'_i = x_i$ with only logarithmic overhead

Hash tree

- $Leaf_i = Hash(File_i)$
- Non-leaf node
  - Hash(Hash(node.left) || Hash(node.right))
- Client only need to store the root
- To prove that a given file is correct, the server only needs to send the client hashes of nodes that follow the path from the given leaf to the root node.