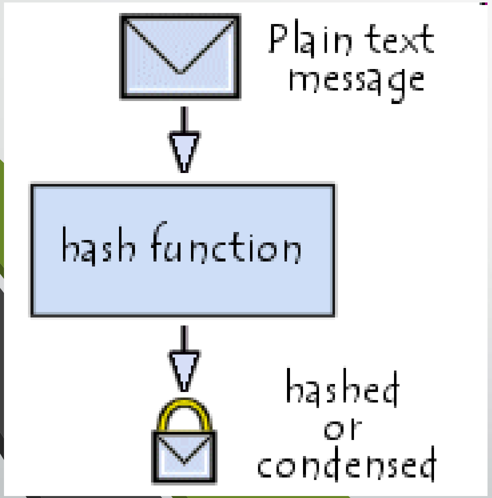
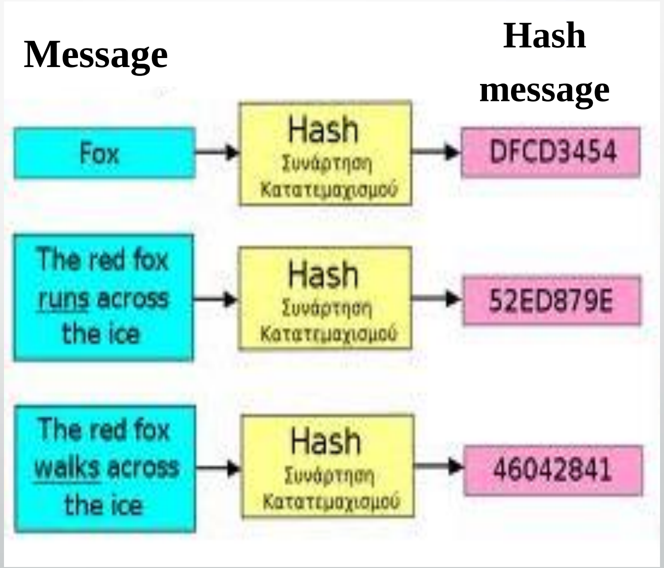
Hash functions

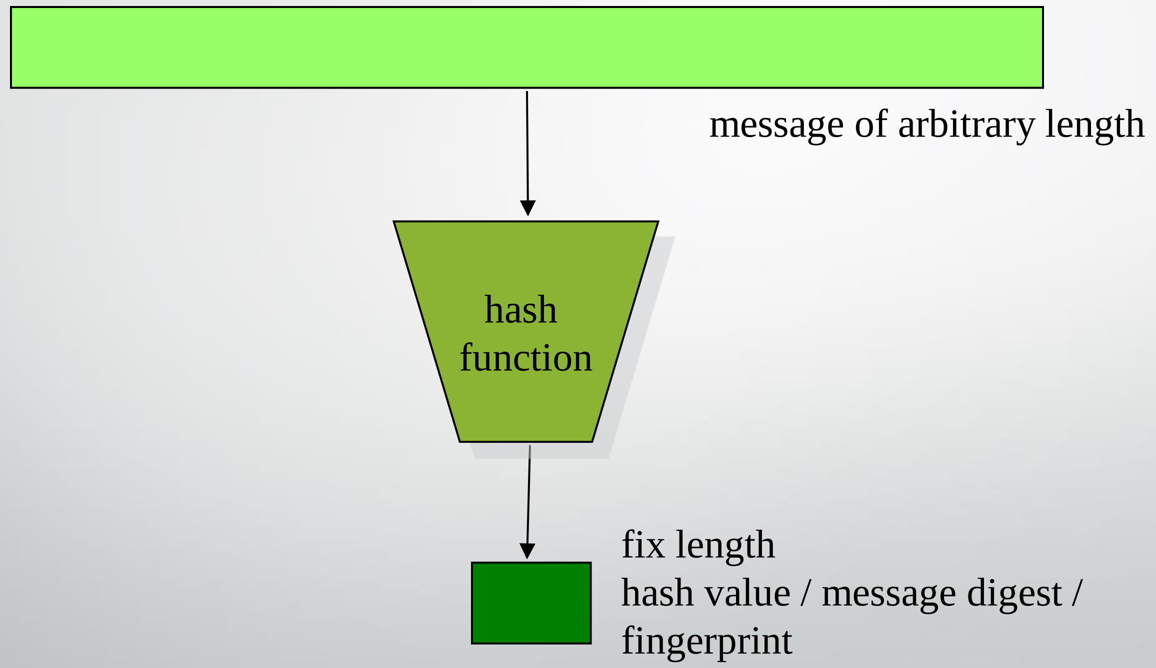




Hash functions

- A hash function maps bit strings of arbitrary finite length to bit strings of fixed length (*n* bits).
- Many-to-one mapping → collisions are unavoidable.
- However, finding collisions are difficult → the hash value of a message can serve as a compact representative image of the message (similar to fingerprints).

Hash functions


message of arbitrary length
hash function
fix length
hash value / message digest / fingerprint

Hash functions

Alice

M

$E_k(M)=C$

h1(M)

Bob

C

$D_k(C)=M$

h1(M)=h2(M)?

Desirable properties of hash functions

- ease of computation
  - given an input *x*, the hash value *h(x)* of *x* is easy to compute;
- **weak collision resistance** (2nd preimage resistance)
  - given an input *x*, it is computationally infeasible to find a second input *x'* such that *h(x') = h(x);*
- **strong collision resistance** (collision resistance)
  - it is computationally infeasible to find any two distinct inputs *x* and *x'* such that *h(x) = h(x');*
- **one-way property** (preimage resistance)
  - given a hash value *y* (for which no preimage is known), it is computationally infeasible to find any input *x* s.t. *h(x) = y*

The Birthday Paradox

- Given a set of N elements, from which we draw k elements randomly (with replacement). What is the probability of encountering at least one repeating element?
- first, compute the probability of no repetition:
  - the first element x_1 can be anything
  - when choosing the second element x_2 , the probability of x_2 $^{1\neq}$ x_1 is 1-1/N
  - when choosing x_3, the probability of x_3 ^{1} x_2 and x_3 ^{1} x_1 is 1-2/N
  - …

The Birthday Paradox

- when choosing the k-th element, the probability of no repetition is 1-(k-1)/N
- the probability of no repetition is (1 - 1/N)(1 - 2/N)…(1 – (k-1)/N)
- when x is small, $(1-x) \approx e^{-x}$
- $(1 - 1/N)(1 - 2/N)\ldots(1 - (k-1)/N) \approx e^{-1/N}e^{-2/N} \ldots e^{-(k-1)/N} = e^{-k(k-1)/2N}$

- the probability of at least one repetition after k drawing is

$$\mathbf{(1 - e^{-k(k-1)/2N})}$$

The Birthday Paradox

- How many drawings do you need, if you want the probability of at least one repetition to be ε ?
- solve the following for k:

$$\varepsilon = 1 - e^{-k(k-1)/2N} \rightarrow k(k-1) = 2N \ln(1/(1-\varepsilon)) \rightarrow$$

$$k \approx \mathrm{sqrt}(2N \ln(1/(1-\varepsilon)))$$

The Birthday Paradox (cont'd)

- examples:

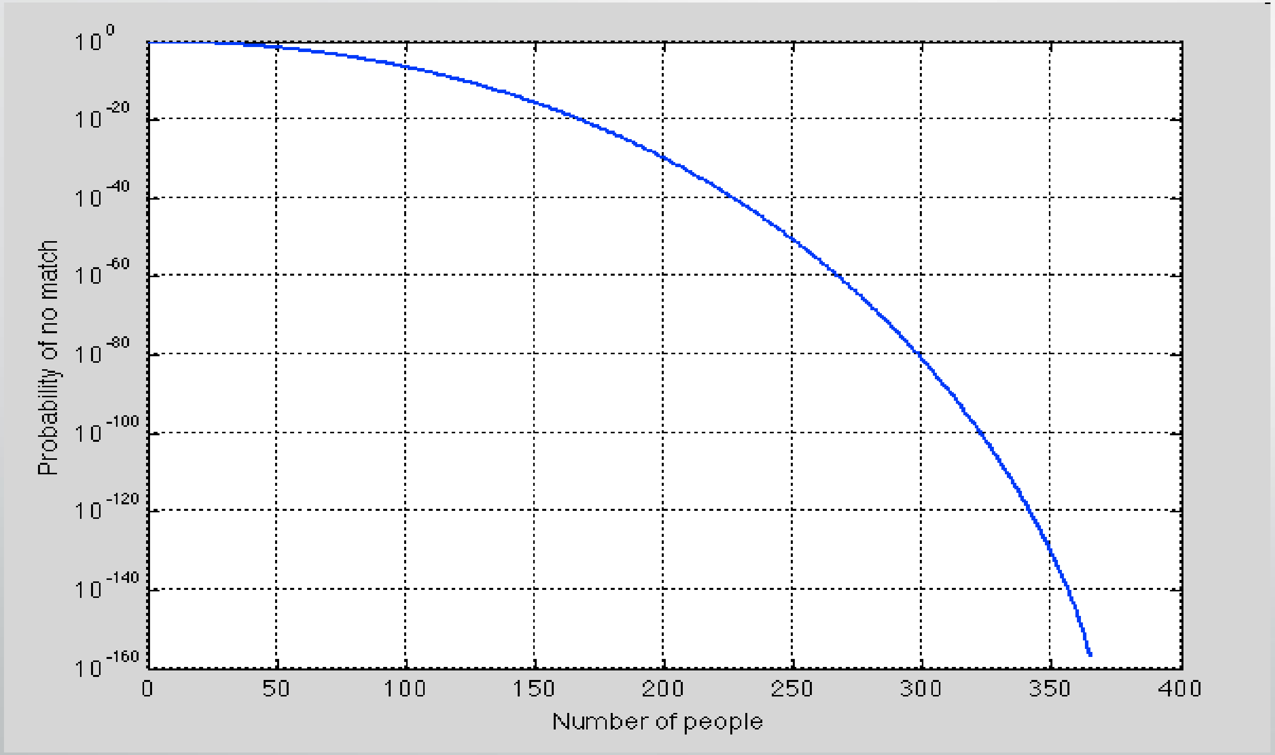
  $\varepsilon = 0.50 \rightarrow k \approx 1.177 \text{ sqrt}(N)$

  $\varepsilon = 0.75 \rightarrow k \approx 1.665 \text{ sqrt}(N)$

  $\varepsilon = 0.90 \rightarrow k \approx 2.146 \text{ sqrt}(N)$

- origin of the name "Birthday Paradox":
  - elements are dates in a year ($N = 365$): among 1.177 sqrt(365) ≈ 23 randomly selected people, there will be at least two that have the same birthday with probability ½

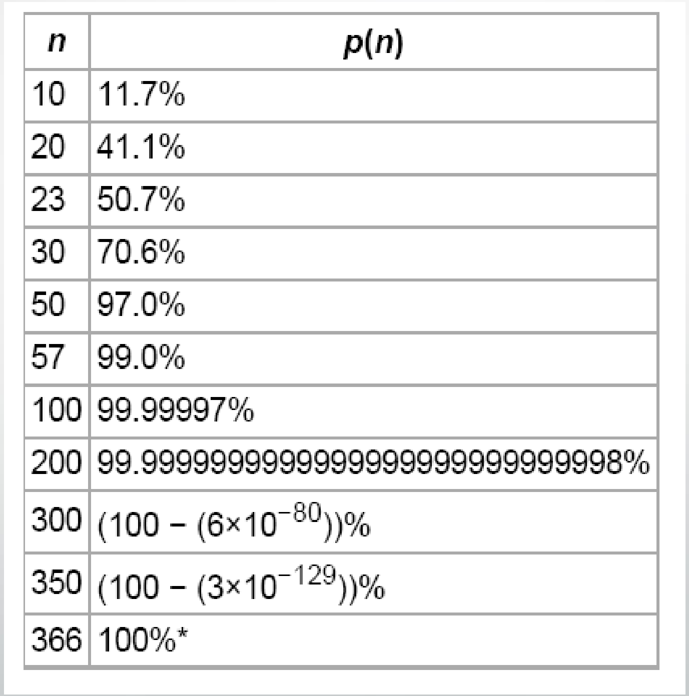
The Birthday Paradox



The Birthday Paradox

The following table shows the probability for some other values of *n*

(This table ignores the existence of leap years):

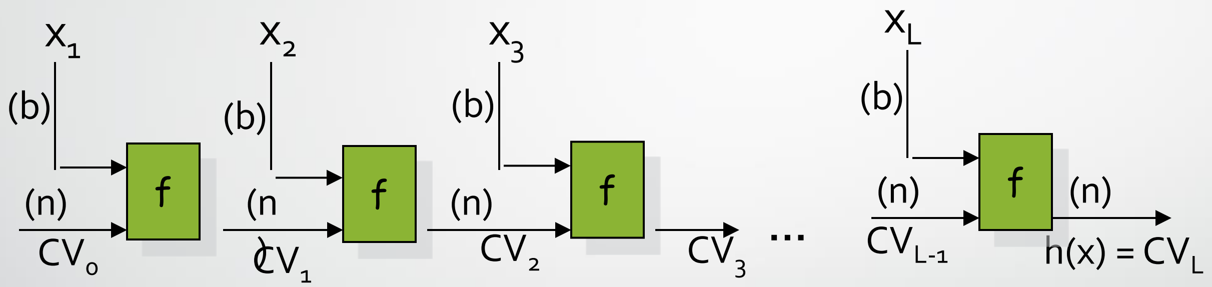


n	*p*(*n*)
10	11.7%
20	41.1%
23	50.7%
30	70.6%
50	97.0%
57	99.0%
100	99.99997%
200	99.9999999999999999999999999998%
300	$(100 - (6\times10^{-80}))\%$
350	$(100 - (3\times10^{-129}))\%$
366	100%*

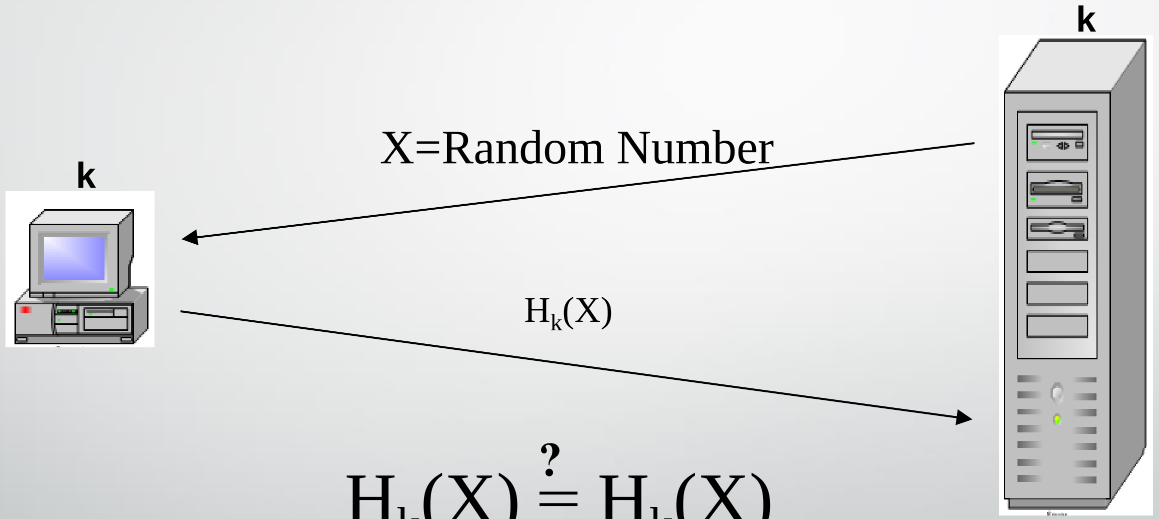
Iterated hash functions

- input is divided into fixed length blocks $x_1, x_2, \ldots, x_L$
- last block is padded if necessary
  - Merkle-Damgard strengthening: padding contains the length of the message
- each input block is processed according to the following scheme; *f* is called the compression function
  - can be based on a block cipher, or
  - can be a dedicated compression function

Iterated hash functions



Authentication via hash function



$$H_k(X) \overset{?}{=} H_k(X)$$