Hash functions

Pierre-Alain Fouque

Introduction

- A Cryptographic hash function is an lagorithm that maps data of arbitrarily size (messages) to fixed-size values (called hash values, digests or hashes)
- It is a one-way function: function hard to invert in practice
- Properties:
  - Deterministic: same message always results in the same hash
  - Quick to compute for any given message
  - Output looks random if we haven’t compute it yet
  - Infeasible to generate a message that yields a given hash values
  - Infeasible to find two different messages with the same hash values
  - A small change to a message should change the hash value (avalanche effect)

Applications

- Cryptographic hash functions have many applications
  - Digital signatures
  - Message authentication codes (MAC)
  - Merkle Tree
  - Password-based hash function competition
  - Bitcoin : blockchain - write new transactions into the blockchain through the mining process

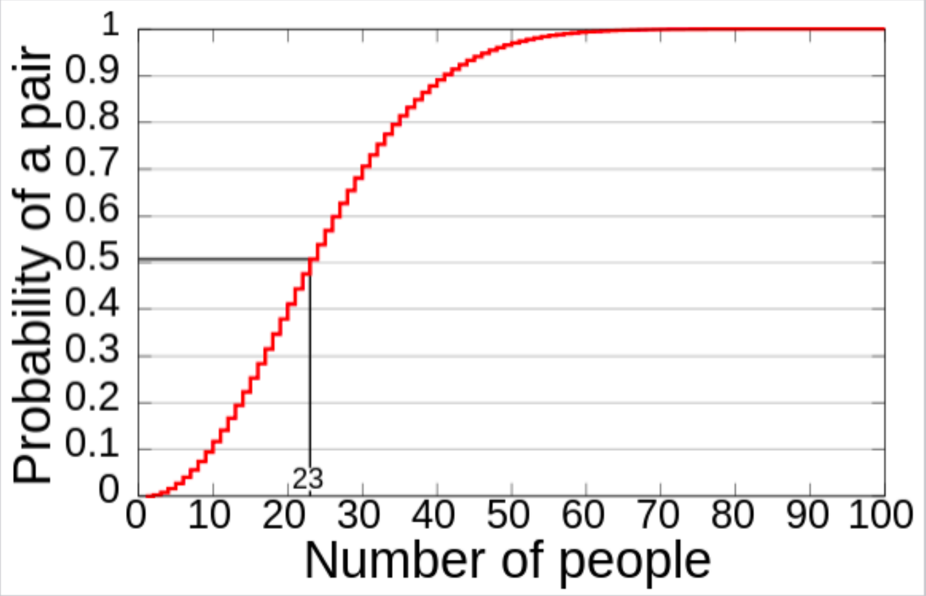
Properties

- Properties: H:{0,1}* to $\{0,1\}^n$
  - Preimages resistance: Given y, find x s.t. H(x)=y
  - Second Preimage resistance: Given y and x s.t. H(x)=y, find x'≠x s.t. H(x')=y
  - Collision: Find x'≠x s.t. H(x')=H(x)

- Attacks:
  - Preimages: Exhaustive search 2^n
    - Compute the hash of random messages x: each hash has a probability $1/2^n$ to match y. On average, the algorithm needs to be executed 2^n times
  - Second Preimage: Exhaustive search 2^n: same analysis
  - Collision: $2^{n/2}$:
    - Compute the hash of random messages and store them in a table. The birthday paradox tells that if the number of messages N is larger than $2^{n/2}$ there will be a collision with probability >1/2.
    - Problem: need to store all (message, hashes) in a table T also $2^{n/2}$ (in memory) and for each message looks through T

Birthday paradox

- In probability theory, the birthday paradox concerns the probability that in a set of n randomly chosen people, some pair of them will have the same birthday
- By the pigeonhole principle, the probability reaches 100% when the number of people reaches 367.
- But, 99.9 % with 70, 50% with 23…



The computed probability of at least two people sharing a birthday versus the number of people

$$\bar{p}(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \cdots \times \left(1 - \frac{n-1}{365}\right)$$

$$= \frac{365 \times 364 \times \cdots \times (365 - n + 1)}{365^n}$$

$$e^{-a/365} \approx 1 - \frac{a}{365}.$$

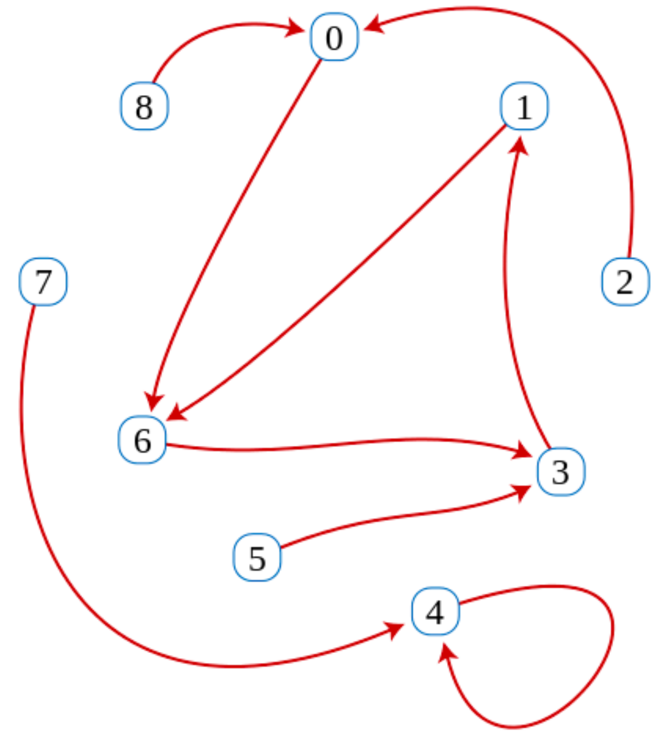
$$p(n) = 1 - \bar{p}(n) \approx 1 - e^{-n(n-1)/730}. \quad p(n) \approx 1 - e^{-n^2/730}$$

Function in a finite set S

- Function from a finite set S to S

- At some point, if we iterate the function and compute the sequence $x_i=f(x_{i-1})$ from x_0 a random element in S, either the sequence will the stationary or we will turn in a cycle

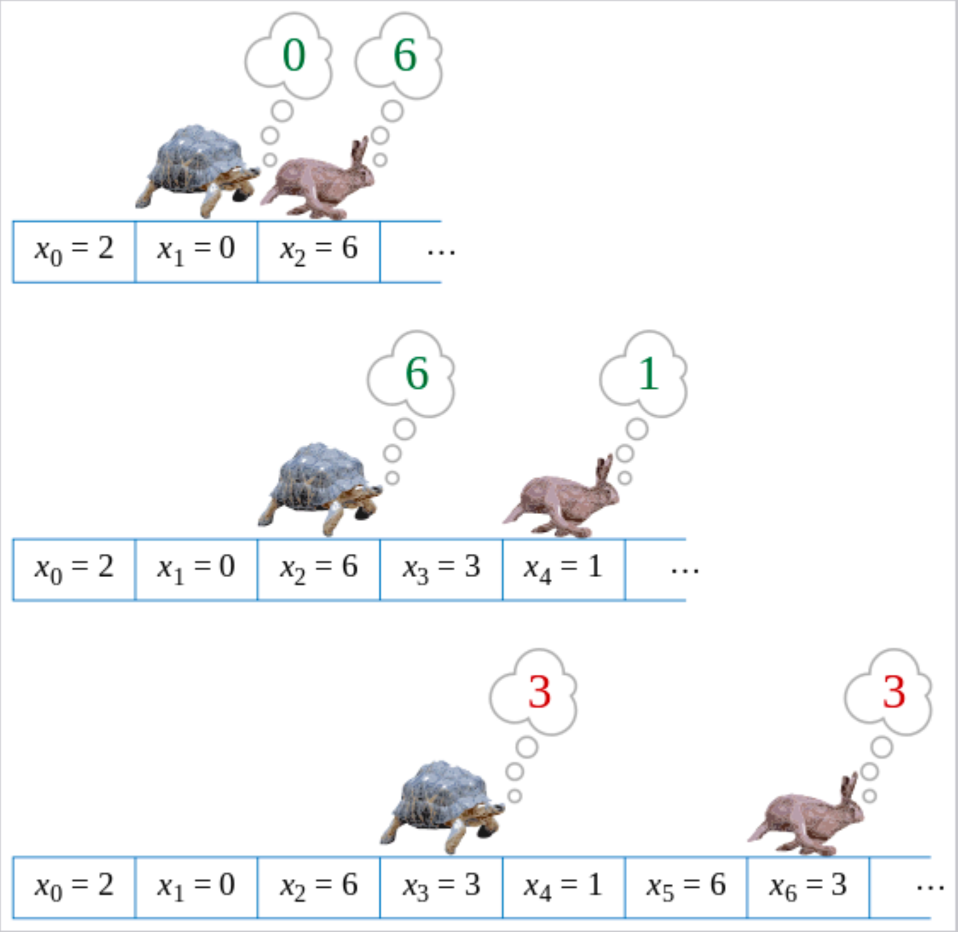
x	$f(x)$
0	6
1	6
2	0
3	1
4	4
5	3
6	3
7	4
8	0



A function from and to the set {0,1,2,3,4,5,6,7,8} and the corresponding functional graph

Looking for collision without memory

- If we are looking for collision, collision will happen in cycle: $x_m = x_{m+l}$ where l (the loop length).
- The cycle detection problem is the task of finding l and m
- Greek letter rho: a path of length m from x_0 to a cycle of length l vertices
- Floyd's Tortoise and Hare algorithm



Floyd's "tortoise and hare" cycle detection algorithm, applied to the sequence 2, 0, 6, 3, 1, 6, 3, 1, ...

Floyd algorithm

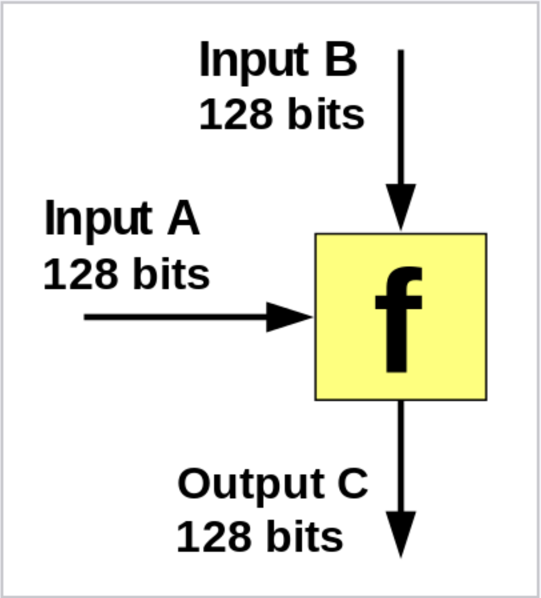
```python
def floyd(f, x0):
    # Main phase of algorithm: finding a repetition x_i = x_2i.
    # The hare moves twice as quickly as the tortoise and
    # the distance between them increases by 1 at each step.
    # Eventually they will both be inside the cycle and then,
    # at some point, the distance between them will be
    # divisible by the period λ.
    tortoise = f(x0) # f(x0) is the element/node next to x0.
    hare = f(f(x0))
    while tortoise != hare:
        tortoise = f(tortoise)
        hare = f(f(hare))
```

```python
# At this point the tortoise position, ν, which is also equal
# to the distance between hare and tortoise, is divisible by
# the period λ. So hare moving in circle one step at a time,
# and tortoise (reset to x0) moving towards the circle, will
# intersect at the beginning of the circle. Because the
# distance between them is constant at 2ν, a multiple of λ,
# they will agree as soon as the tortoise reaches index μ.

# Find the position μ of first repetition.
mu = 0
tortoise = x0
while tortoise != hare:
    tortoise = f(tortoise)
    hare = f(hare)   # Hare and tortoise move at same speed
    mu += 1
```

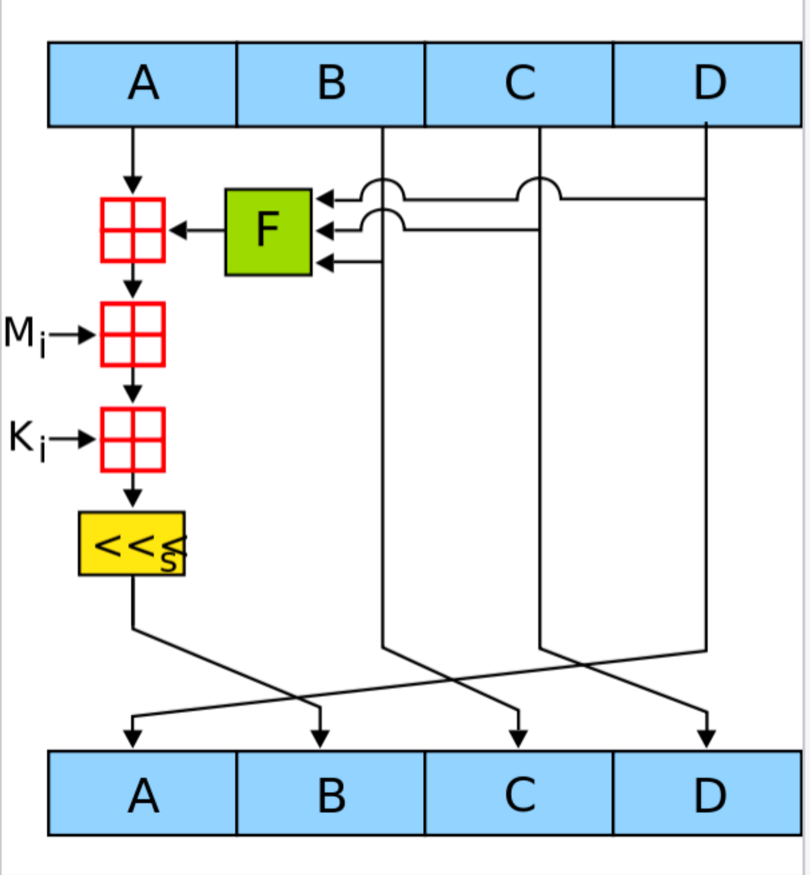
```python
# Find the length of the shortest cycle starting from x_μ
# The hare moves one step at a time while tortoise is sti
# lam is incremented until λ is found.
lam = 1
hare = f(tortoise)
while tortoise != hare:
    hare = f(hare)
    lam += 1

return lam, mu
```

Compression function

- f:$\{0,1\}^m$ vers $\{0,1\}^n$ with m>n
- Same properties has hash function
- Other constructions:
  - $f(m,h) = E_h(m) \oplus h$?
  - $f(m,h) = E_m(h)$? $f(m,h) = E_h(m)$?
- Ex: $f(m,h) = E_m(h) \oplus h$ (Davies-Meyer)



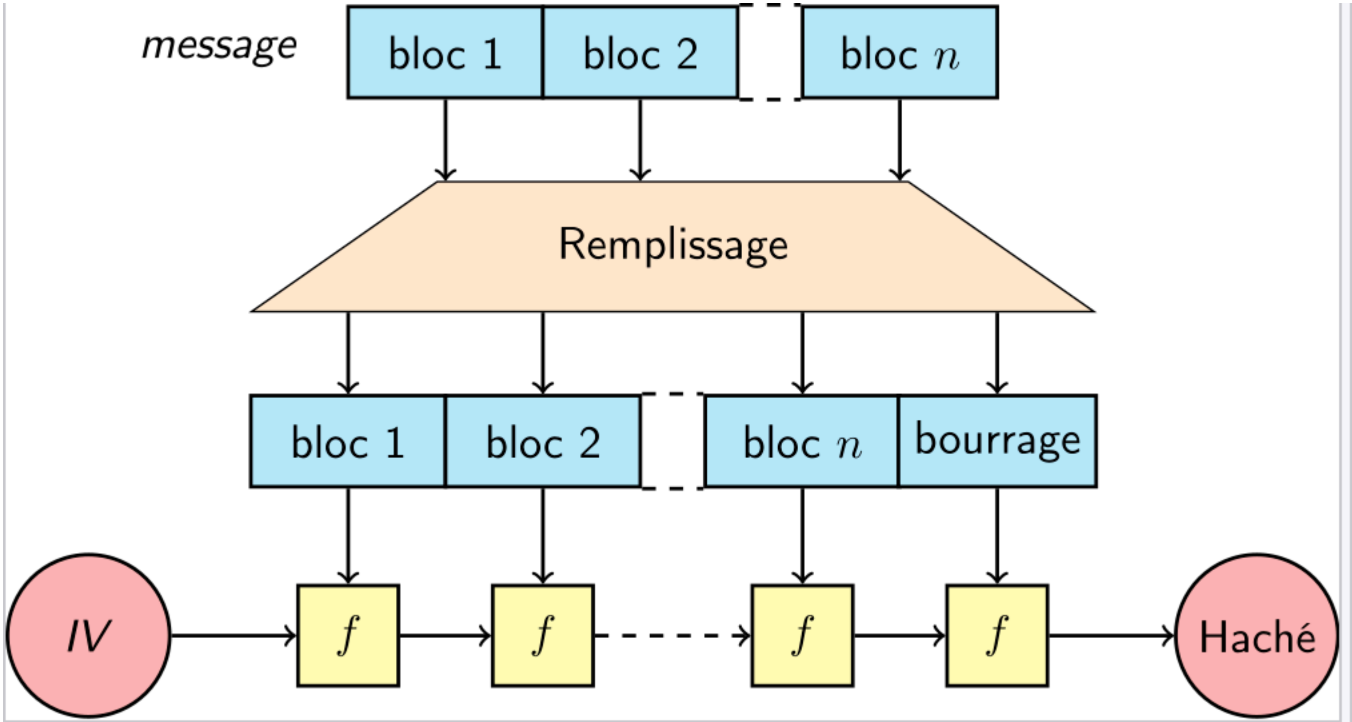
A one-way compression function



One MD4 operation : MD4 consists of 48 of these operations, grouped in three rounds of 16 operations. *F* is a nonlinear function; one function is used in each round. M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each round.

Merkle-Damgard: From compression function to hash function

- Domain Extension Technique:
- Thm: If the compression function f is collision resistant, so is the hash function built from f using the MD construction
- Padding is important and contains the size of the message in bits



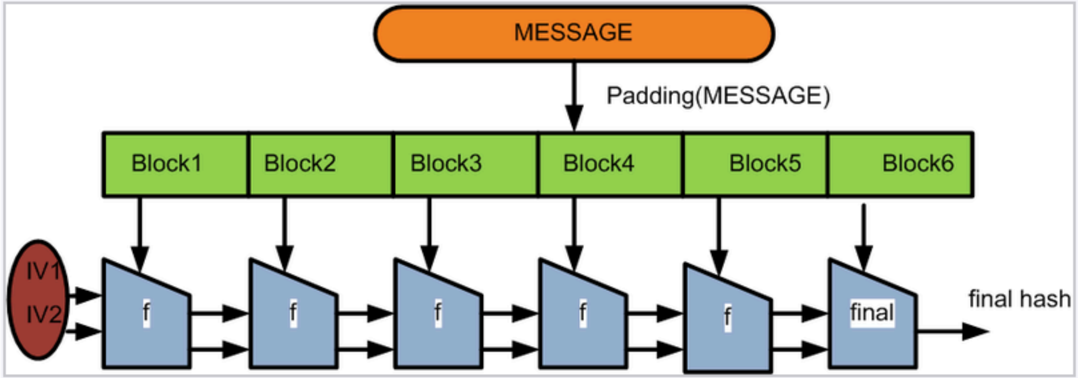
Construction de Merkle-Damgård avec la fonction de compression *f* et le vecteur d'initialisation *IV* et un schéma de remplissage

Problems with Merkle-Damgard: multicollision

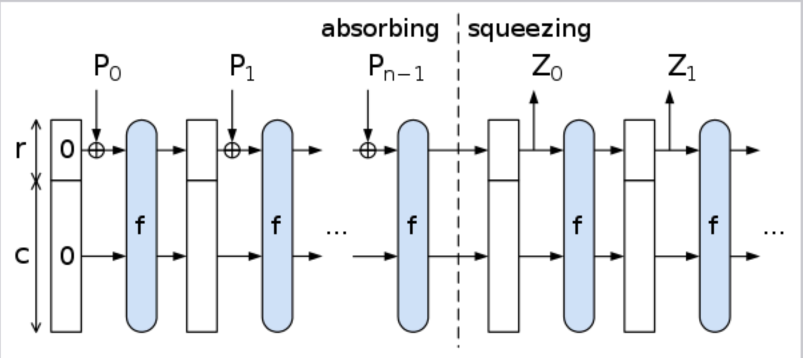
- Merkle-Damgard used during more than 15 years without problems
- In 2004, Joux discovered multicollision when studying the resistance of the hash function constructed in some RFC: H(m)=SHA1(m)||MD5(m)
- If the MD5 is broken, maybe SHA-1 will not be, or it is unexpected that the same message will collide for both …
- Resistance of H against collisions is : $2^{(160+128)/2}=2^{144}$ since the output size of SHA-1 is 160 bits and MD5 is 128
- Multicollsion: generate many messages with same hash
  - For random function it will be $2^{(k-1)/kn}$ for k messages with n bit of outputs
  - Surprisingly, $k.2^{n/2}$ for MD constructions …
  - Then, easy to break H with $2^{80}+80.2^{64}$ hash computations

Wide-pipe and SHA-3

- Wide-pipe construction to avoid multicollisions: double the internal size of the function
- NIST proposes a new competiton in 2005 and the winner is Keccak with a new design
- Sponge Constructions
- SHAKE: Pseudo-Random Generator with arbitrary size output



The Wide pipe hash construction. The intermediate chaining values have been doubled.



The sponge construction for hash functions. P_i are input, Z_i are hashed output. The unused "capacity" c should be twice the desired resistance to collision or preimage attacks.