Cryptographic hash function

$H(x) \rightarrow y$

string
byte array

fixed size byte array

- lock random — Hashing something new, gives a random value
- be deterministic — Hashing something twice gives the same value

Properties

Pre-image resistance

given y cannot find x
$H(x) = y$

Weak collision resistance

given x cannot find x'
s.t. $H(x) = H(x')$

Collision resistance

cannot find x x' s.t.

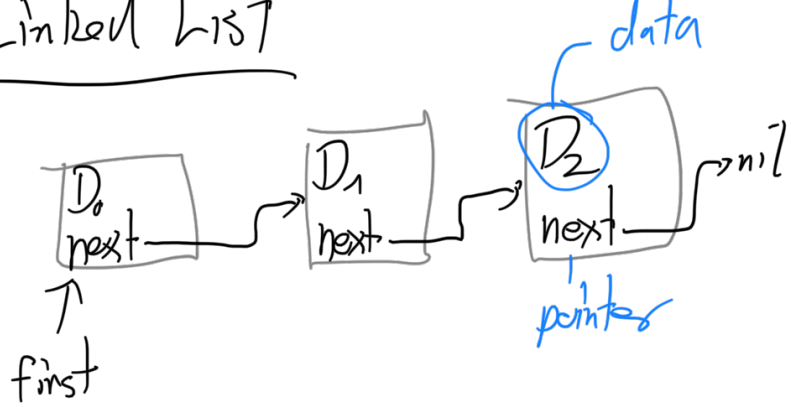
$$H(X) = H(X')$$

Example use:

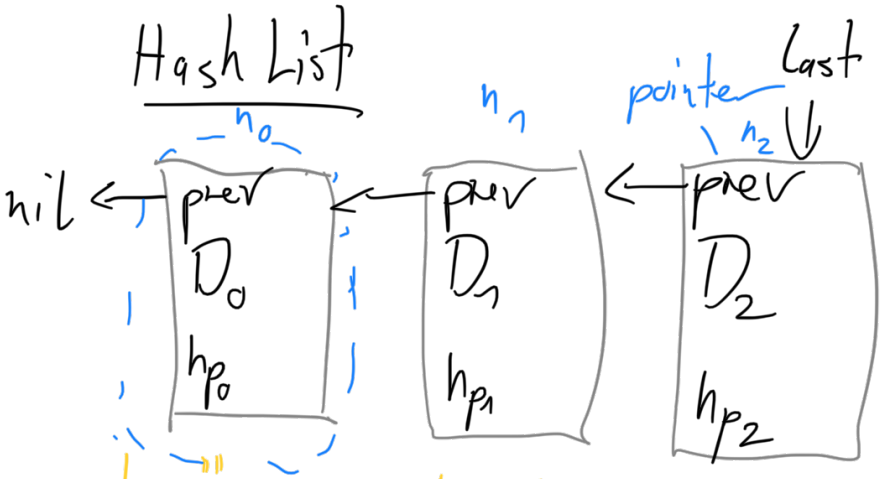
- Hashing passwords.
- Checksums e.g. in HTML source tags.

Hash Chain

Linked List



Hash List

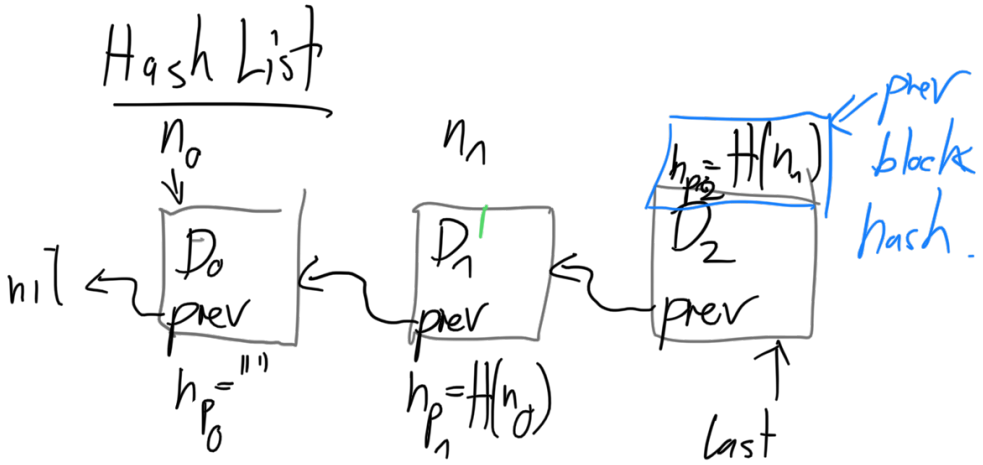


$h_2 = H(h_1) = H(D_1 \| h_{p_1})$ ← gives an id for h_1

if we change D_1 to D_1'

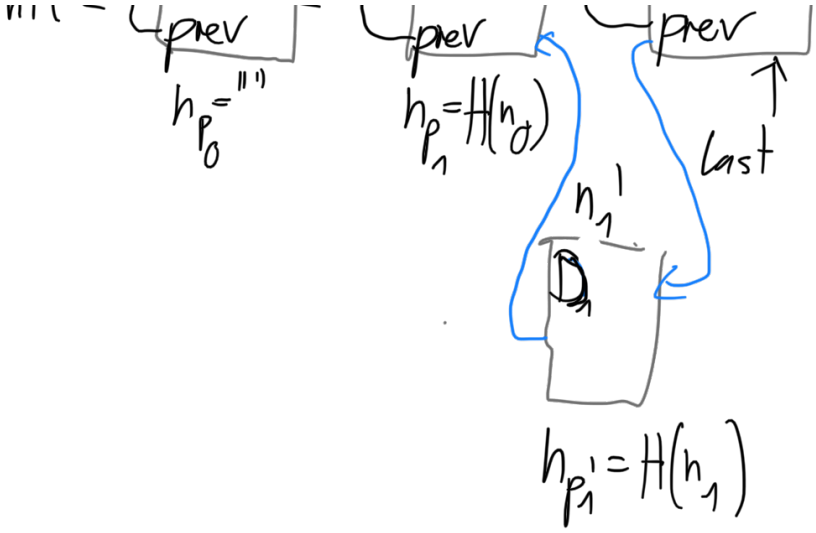
$\rightsquigarrow H(n_1)$ changes

$\rightsquigarrow h_{p_2}$ is no longer correct.



$H(D_1' \| h_{p_1}) \neq h_{p_2}$

if we insert a block, h_{p_2} is no longer correct.



$$H(h_1') = H(D_1 \| h_{p_1'})$$

h_{p_2} $\qquad$ h_{p_1}

If we know the last hash,
we can retrieve the chain from some
cloud service and check that it is correct.

Example use

store all passport numbers
issued on one day in one block.

→ cannot take old passports.

→ Idea: Add timestamps to the blocks.

Merkle Trees

From Example: can we check that a number is in the block, without knowing all Numbers.

Numbers are data items

Plaintext Data:

$$D_i = N_1 \| N_2 \| N_3 \| N_4 \ldots$$

long and data is public

Data is a hash

$$D_i = H(N_1 \| N_2 \| N_3 \| \ldots)$$

short, data is not public.

but to check if one number is these, I need all the numbers.

Data is hashes

$$D_i = H(N_1) \| H(N_2) \| H(N_3) \| \ldots$$

data is not public (but can guess a correct number)
can check inclusion.

Data is hash of hashes

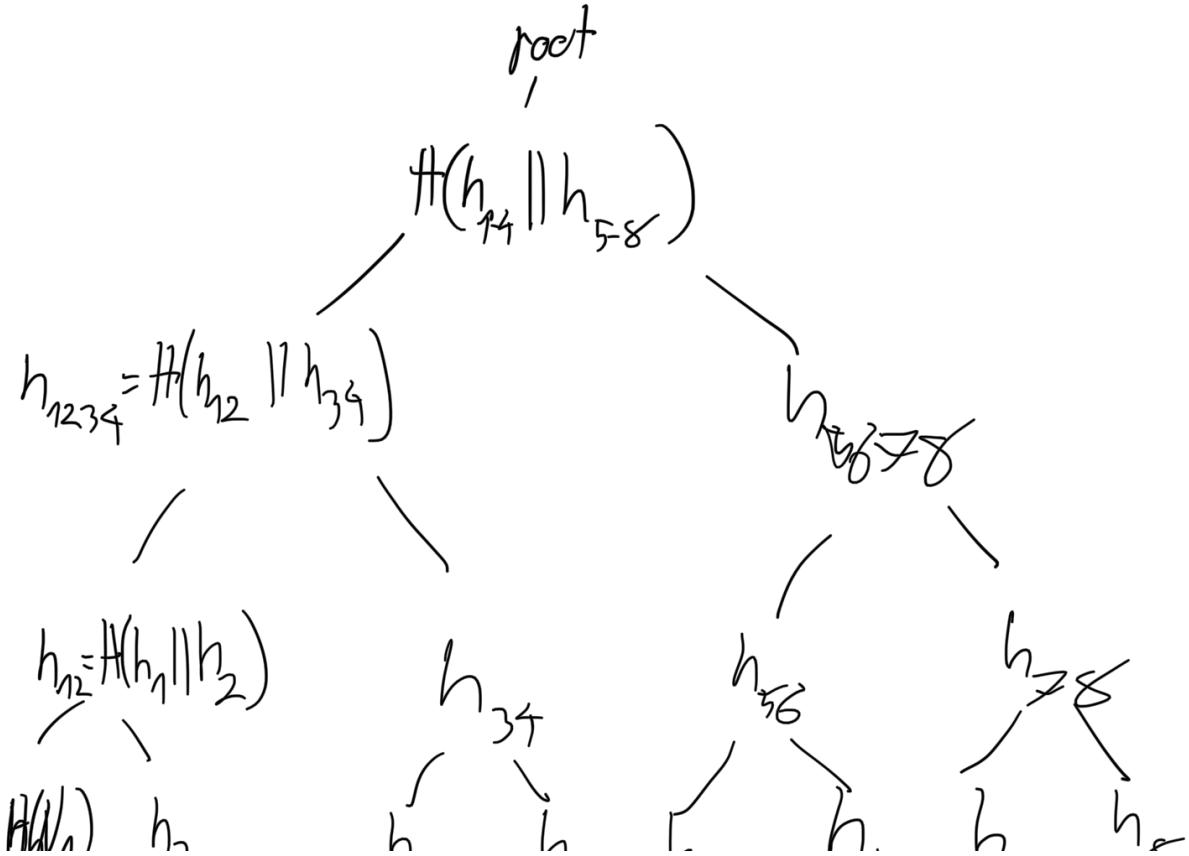
$$D_i = H\left(H(N_1) \| H(N_2) \| H(N_3) \| \dots\right)$$

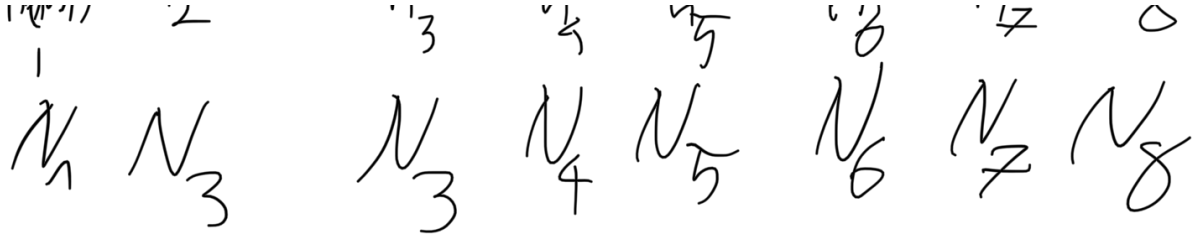
data is not public

can check inclusion if I know N_i and $\forall_i\ H(N_i)$

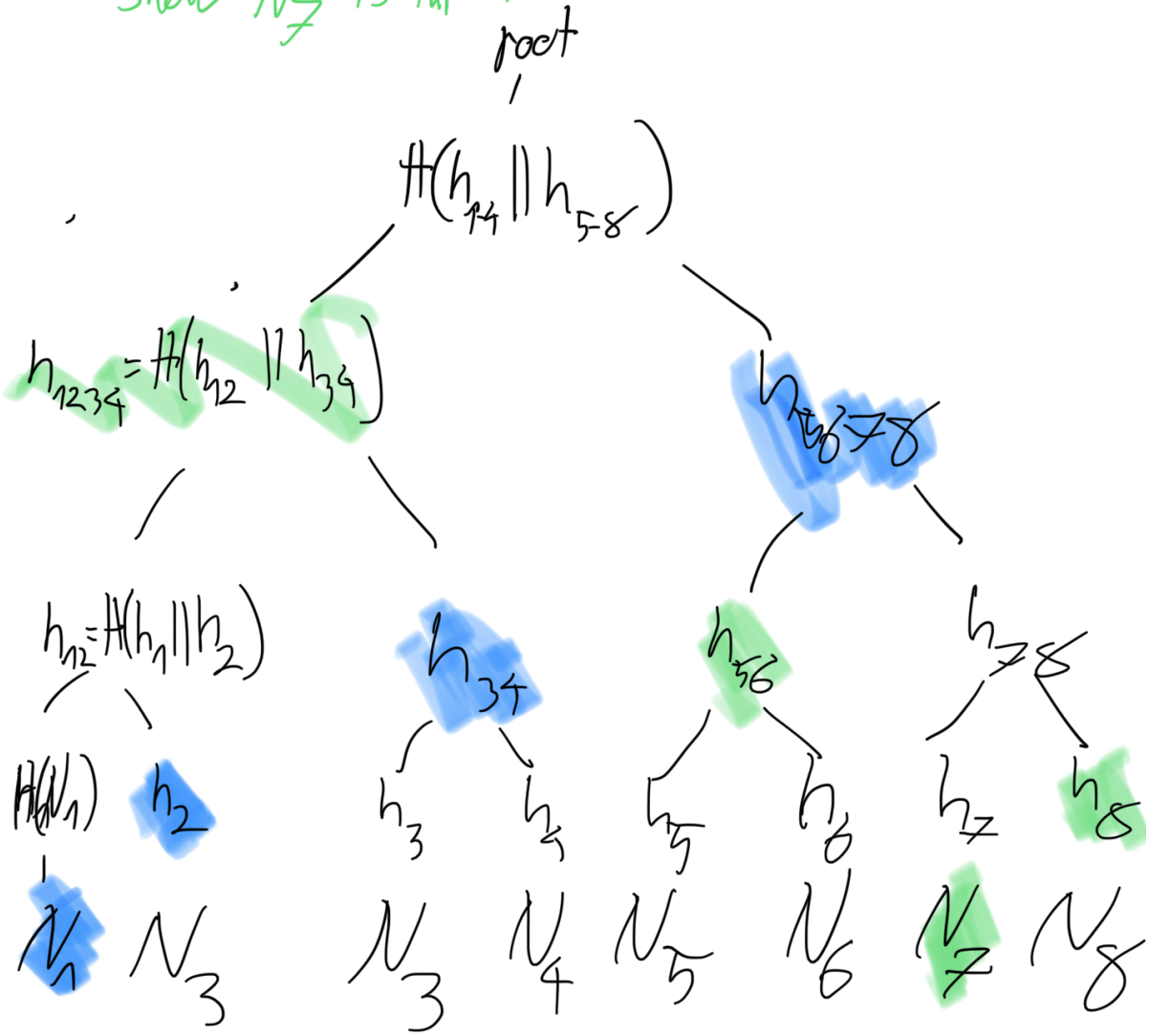
(fewer targets for guessing)

Merkle tree





To show N_1 is in the tree, we need:
show N_7 is in the tree



Proof for N_1

N_1 h_a, h_b, h_c (left, left, left)

$$root \stackrel{?}{=} H(H(H(H(N_1) \| h_a)) \| h_b) \| h_c)$$ [0,0,0]

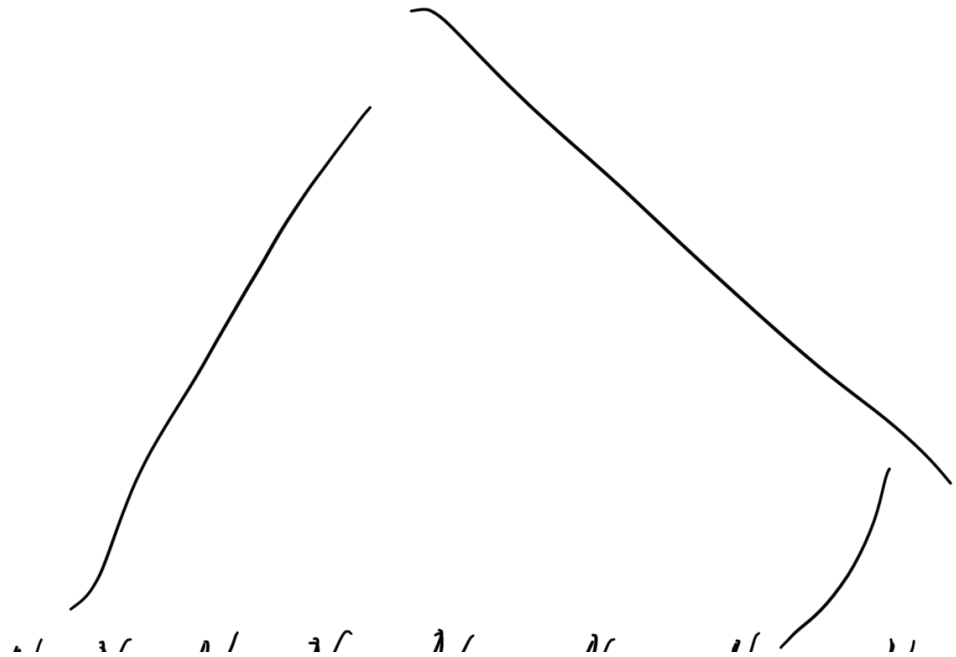
Proof for N_7

N_7, h_a (h_8), h_b (h_{56}), h_c (h_{14}), left, right, right

$$root \stackrel{?}{=} H(h_c \| H(h_b \| H(H(N_7) \| h_a)))$$

What if I only have 5 Data items?

- duplicate N_5
- add default element



N_1 N_2 N_3 N_4 N_5 None None None