Revisiting the Long Message Attack

and Auditing its cost

Ahmed Alharbi

Sorbonne Univertisté

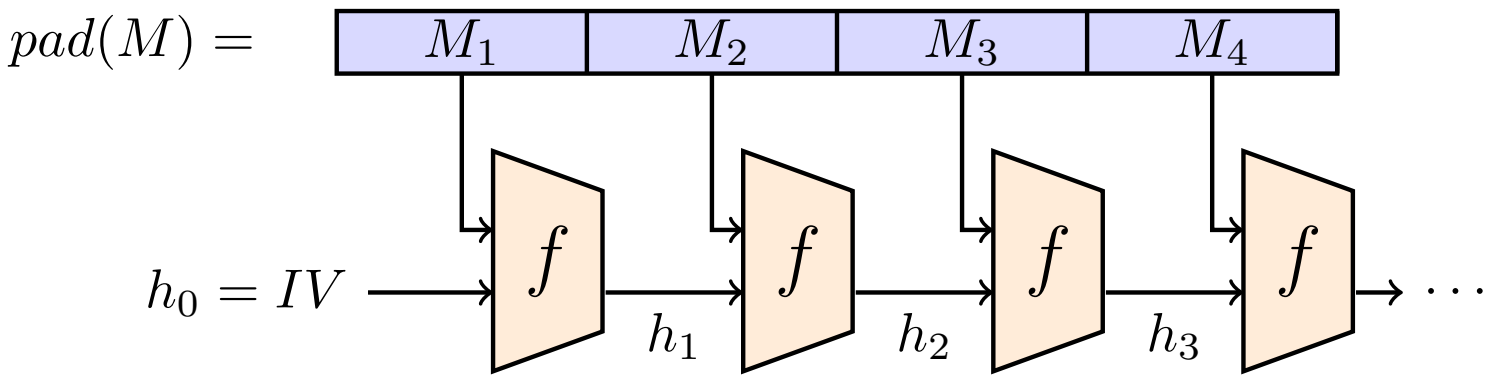
ahmed.alharbi@lip6.fr

Journées C2
16 Oct 2023

Review Hash Functions

Definition

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



It's easy to compute $H(M||L)$ given $H(M)$

`SHA-1, SHA-2, MD5`

2^{nd}-preimage

- $H : \{0,1\}^* \rightarrow \{0,1\}^n$
- Given, M and $H(M)$ find L s.t. $H(L) = H(M)$
- For a random L:

$$Pr\left[H(M) = H(L)\right] = \frac{1}{2^n}$$

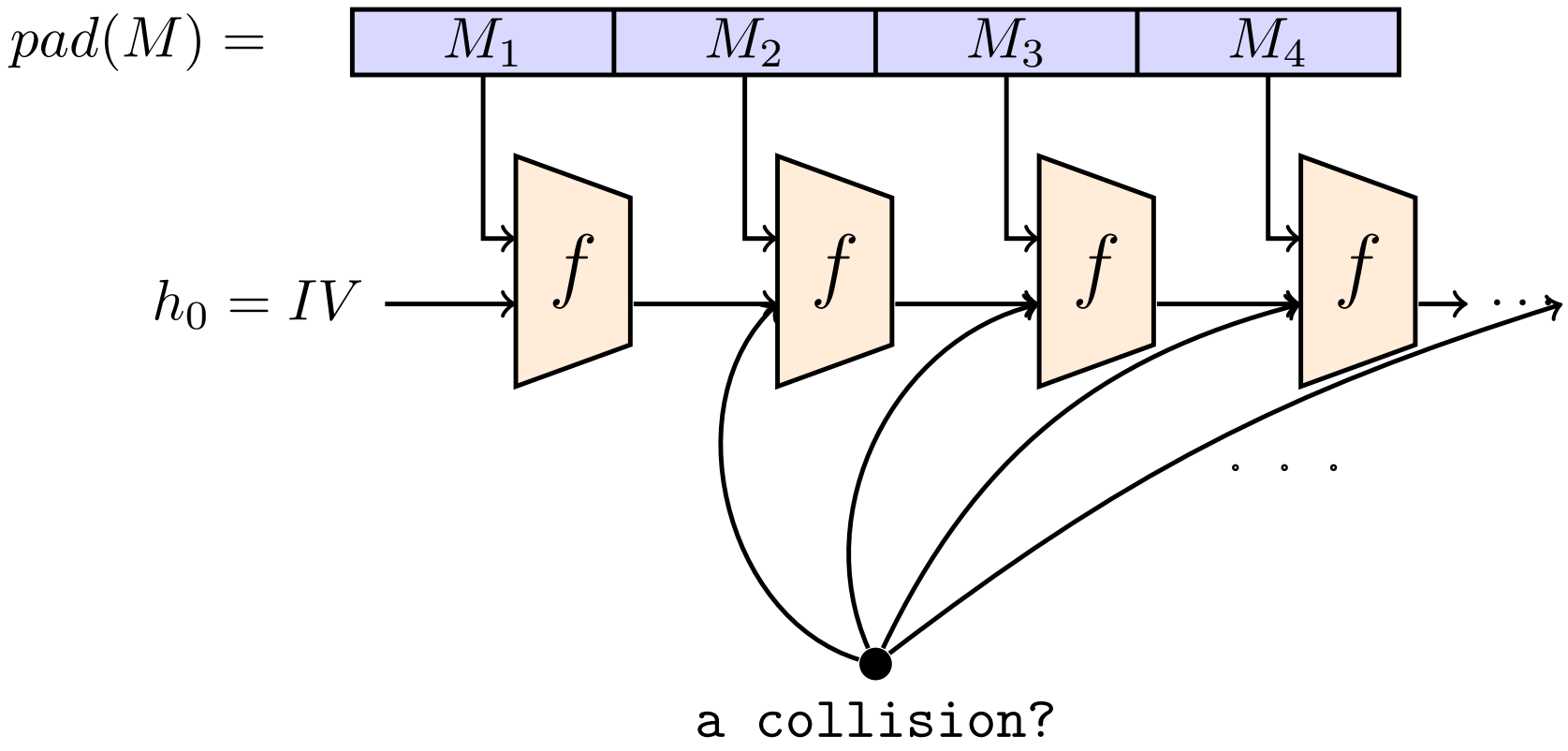
2^{nd}-preimage

- $H : \{0,1\}^* \rightarrow \{0,1\}^n$
- Given, M and $H(M)$ find L s.t. $H(L) = H(M)$
- For a random L:

$$Pr[H(M) = H(L)] = \frac{1}{2^n}$$

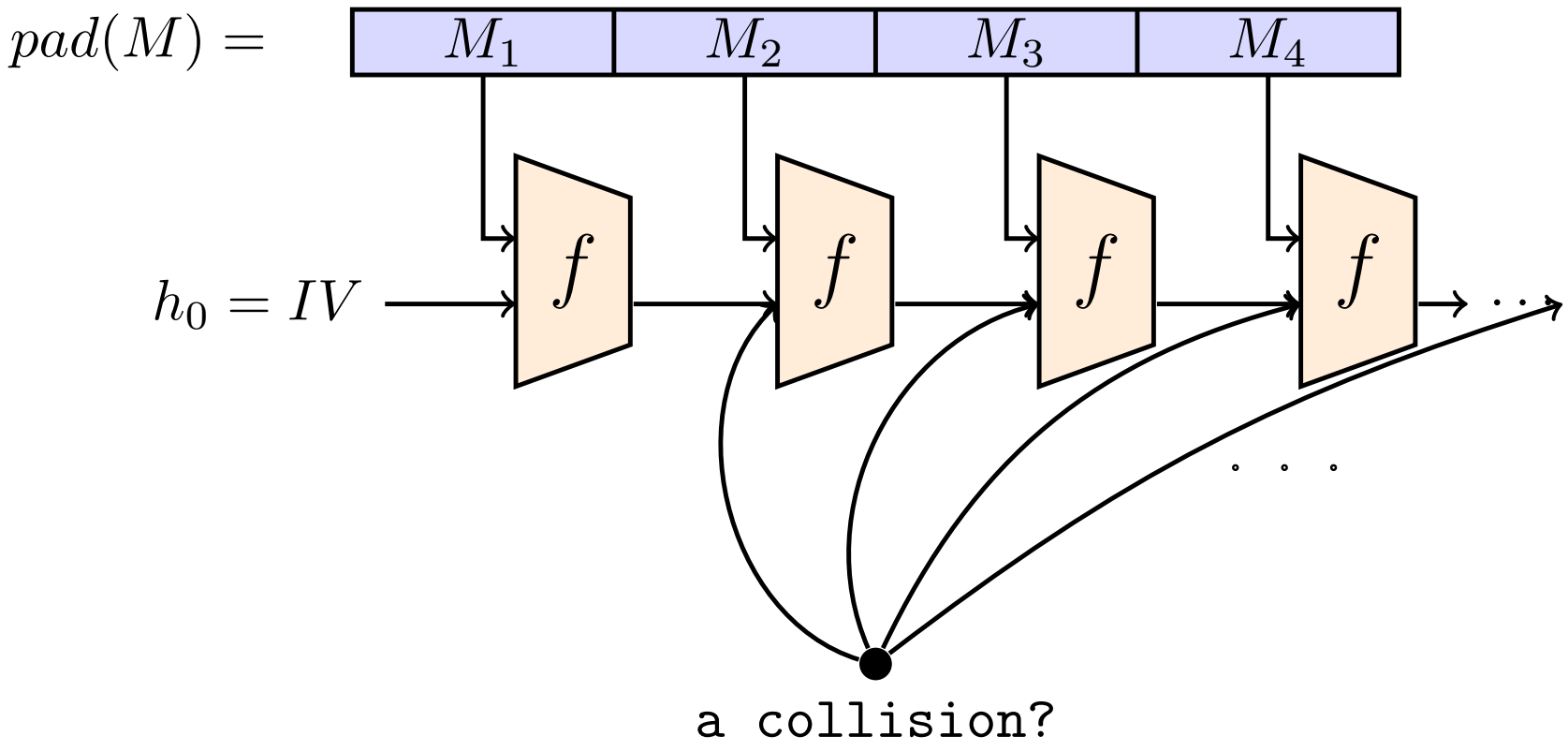
- On average 2^n different L to get a second-preimage.

2^{nd}-preimage



If M has k blocks, then $Pr[H(\cdot) = h_i] = \frac{k}{2^n}$, thus we need around $\frac{2^n}{k}$ trials.

2nd-preimage



If M has k blocks, then $Pr[H(\cdot) = h_i] = \frac{k}{2^n}$, thus we need around $\frac{2^n}{k}$ trials.

Cost* : $\underbrace{k}_{\text{generate M}} + \underbrace{\frac{2^{n/2}}{k}}_{\text{random messages}}$, optimal $k := 2^{n/2}$

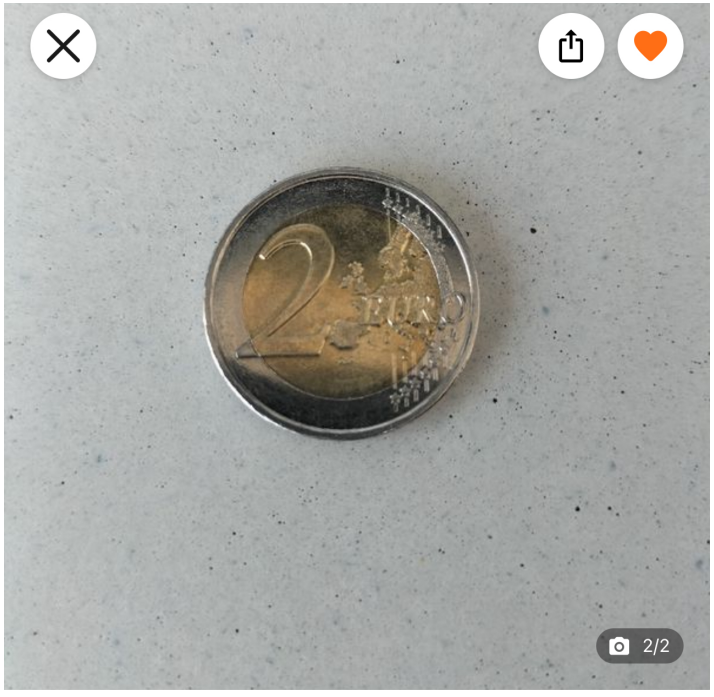
n=96 and l=48, i.e. $|M| = 2^{48}$ blocks

n=96 and l=48, i.e. $|M| = 2^{48}$ blocks

with `sha_ni` generates
$\approx 2^{24} hash/sec \Rightarrow$ we need
$2^{48-24} sec \approx 4660$ *hours* $\approx$ **140** €

n=96 and l=48, i.e. $|M| = 2^{48}$ blocks

with `sha_ni` generates
$\approx 2^{24} hash/sec \Rightarrow$ we need
$2^{48-24} sec \approx 4660\ hours \approx \mathbf{140}\ €$

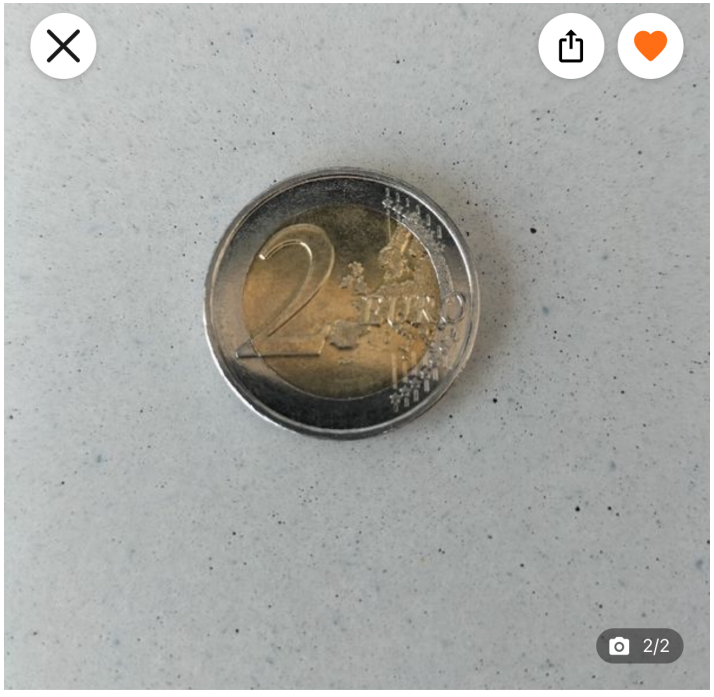


Pièces rare 1987

150 € Livraison : à partir de 3,49 €

n=96 and l=48, i.e. $|M| = 2^{48}$ blocks

with `sha_ni` generates
$\approx 2^{24} hash/sec \Rightarrow$ we need
$2^{48-24} sec \approx 4660$ *hours* $\approx$ **140** €



However,

- We need around `3377 TB` to store 2^{48} hashes.
- The largest memory we could get is `98.3 TB` across 512 servers!
- Each server has `192GB` Memory

Distinguished point

Only store h_i that ends with d-`zeros`

- Memory usage is $2^{-d} \times$ *before*
- Increases the `CPU` work by the factor 2^d

Distinguished point

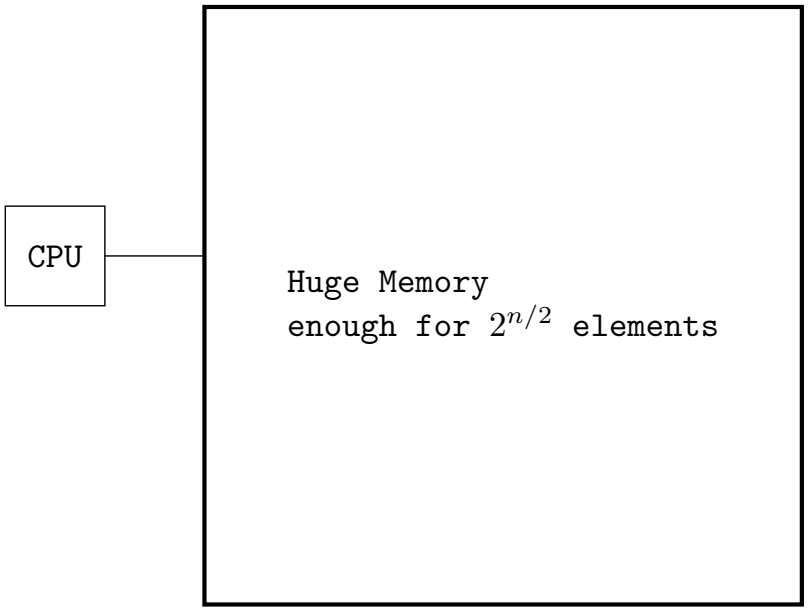
Only store h_i that ends with d-`zeros`

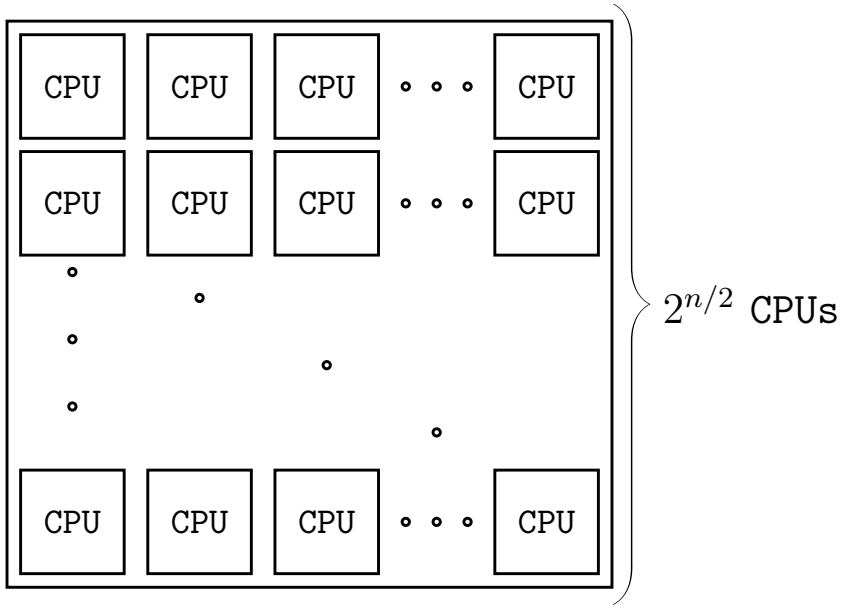
- Memory usage is $2^{-d} \times$ *before*
- Increases the `CPU` work by the factor 2^d

Or even better, with above increase the long message size by 2^x

- Memory usage is $2^{x-d} \times$ *before*.
- Increases `CPU` work by the factor 2^{d-x}
- Sweet spot $x := \frac{d}{2}$

Models of Computations



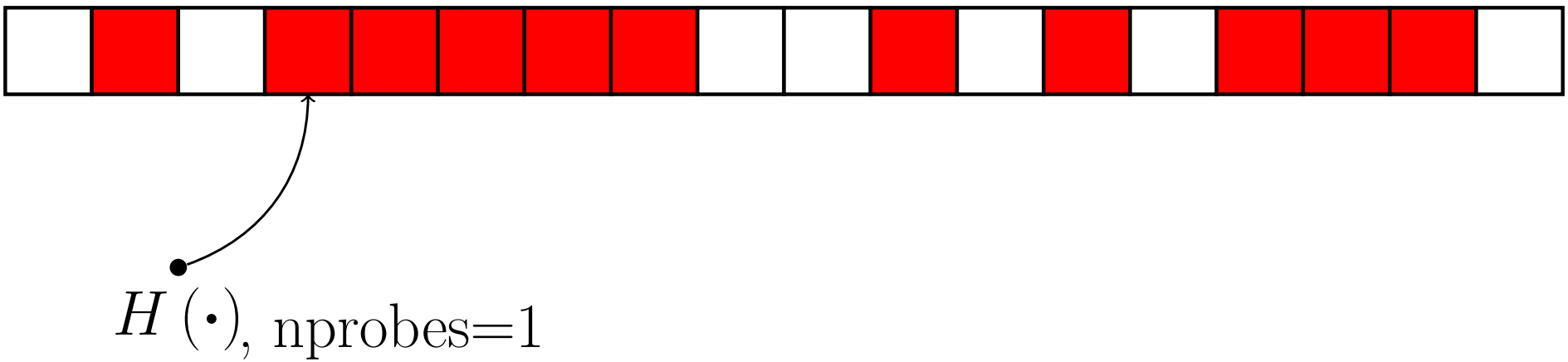


- a 2^{nd}-preimage is found after a CPU generates $2^{n/2}$ random digests and memory accesses.
- # operations = $\underbrace{2^{n/2}}_{\text{CPU}} + \underbrace{2^{n/2}}_{\text{memory}}$

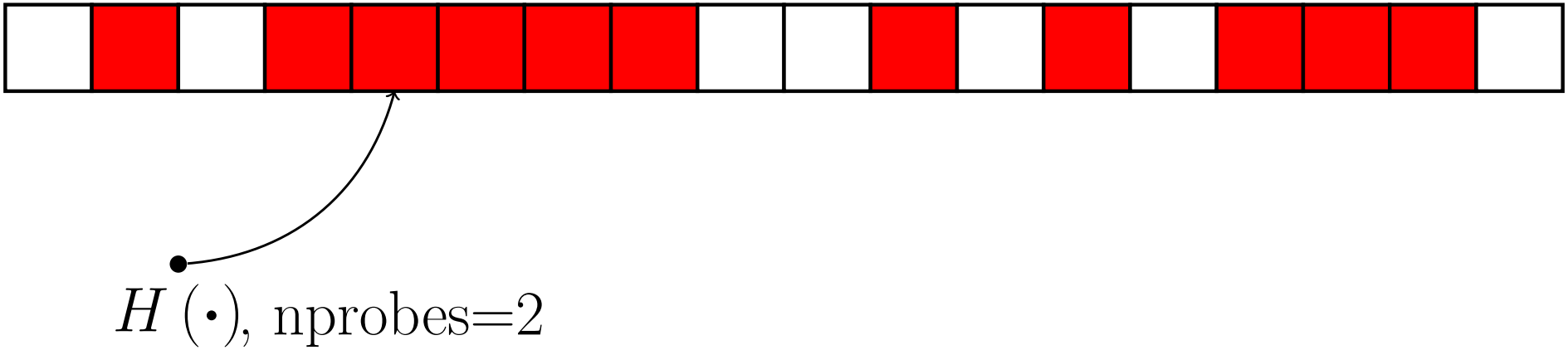
- a 2^{nd}-preimage is found after a CPU generates $2^{n/2}$ random digests.
- #operations = $\underbrace{2^{n/2}}_{\#\,\text{CPUs}} \times \underbrace{2^{n/2}}_{\#\text{operations/CPU}} = 2^n$

Could we reduce our memory footprint further?

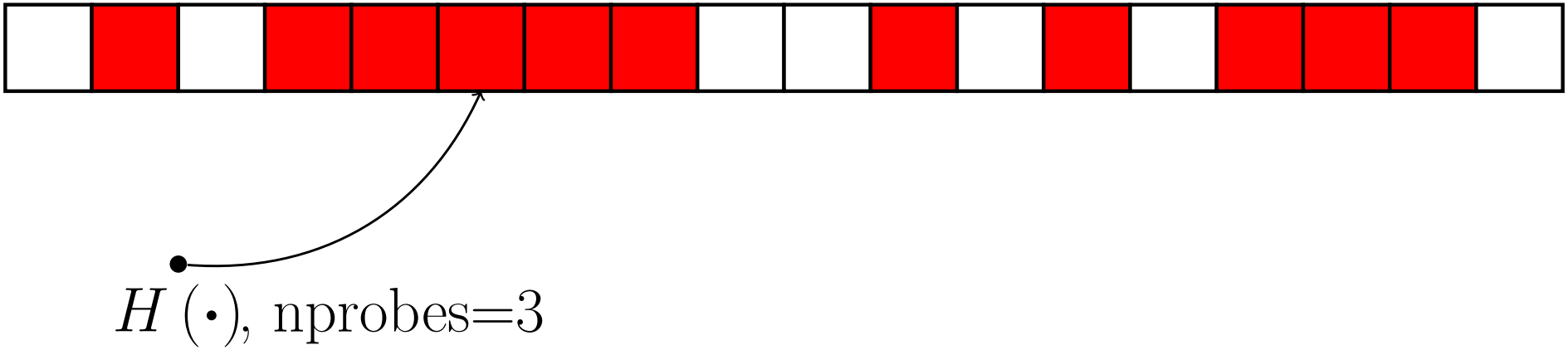
Linear Probing



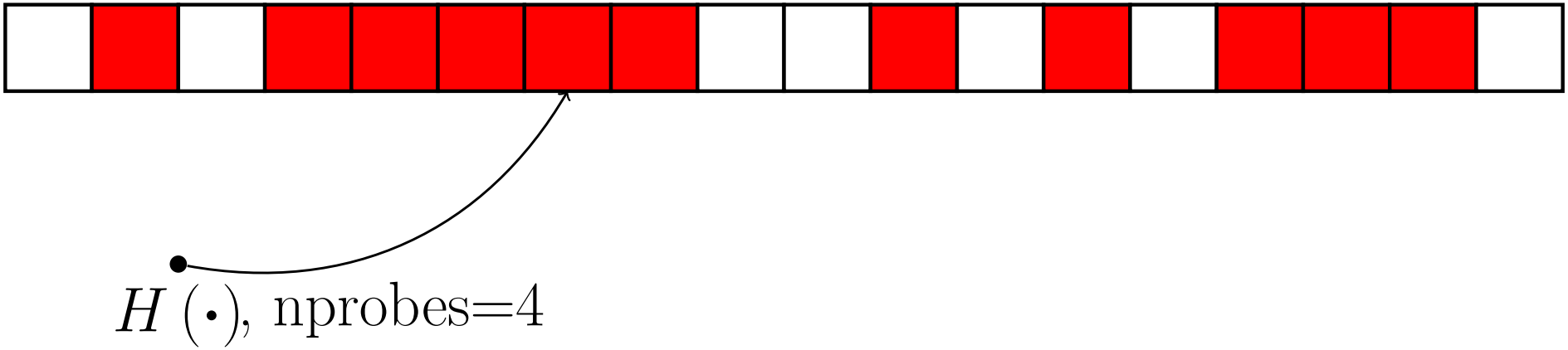
Linear Probing



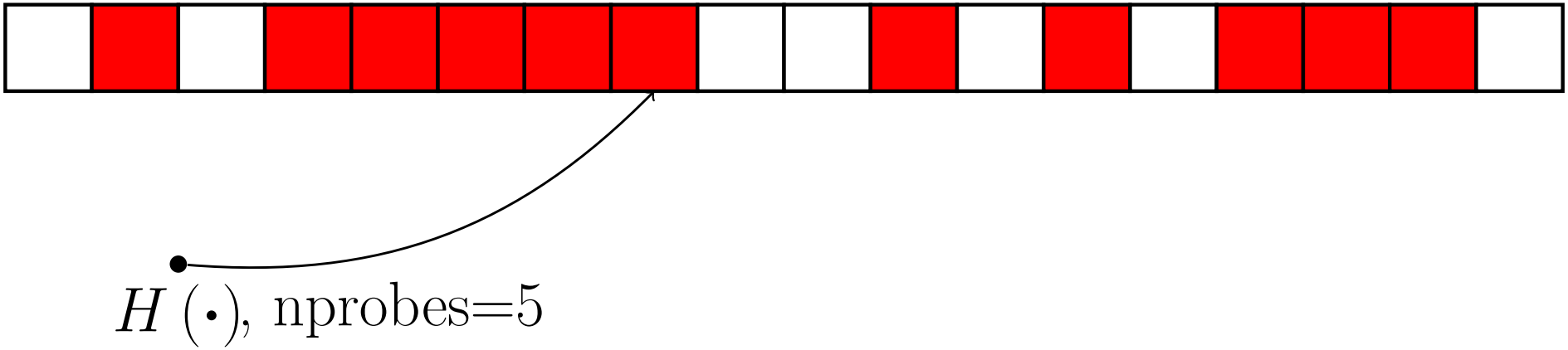
Linear Probing



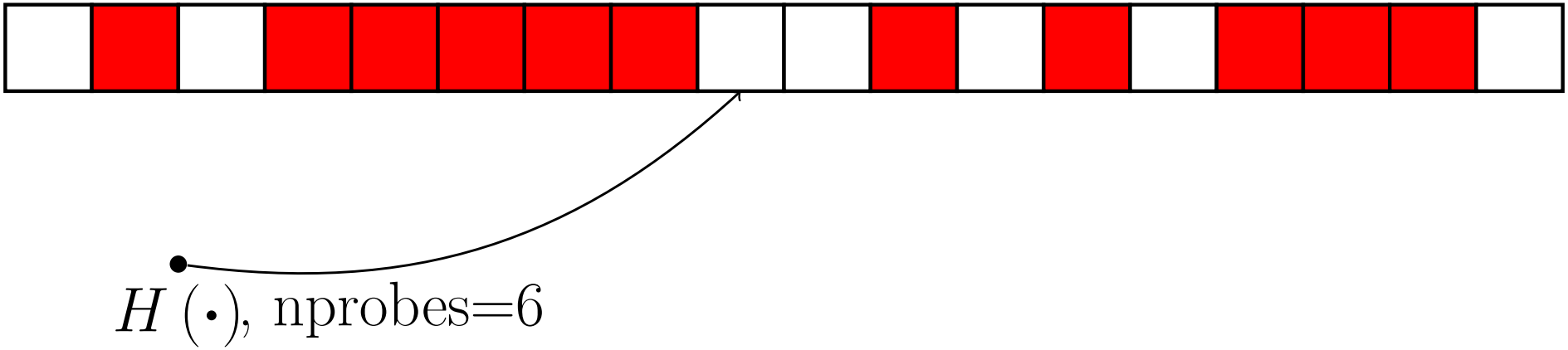
Linear Probing



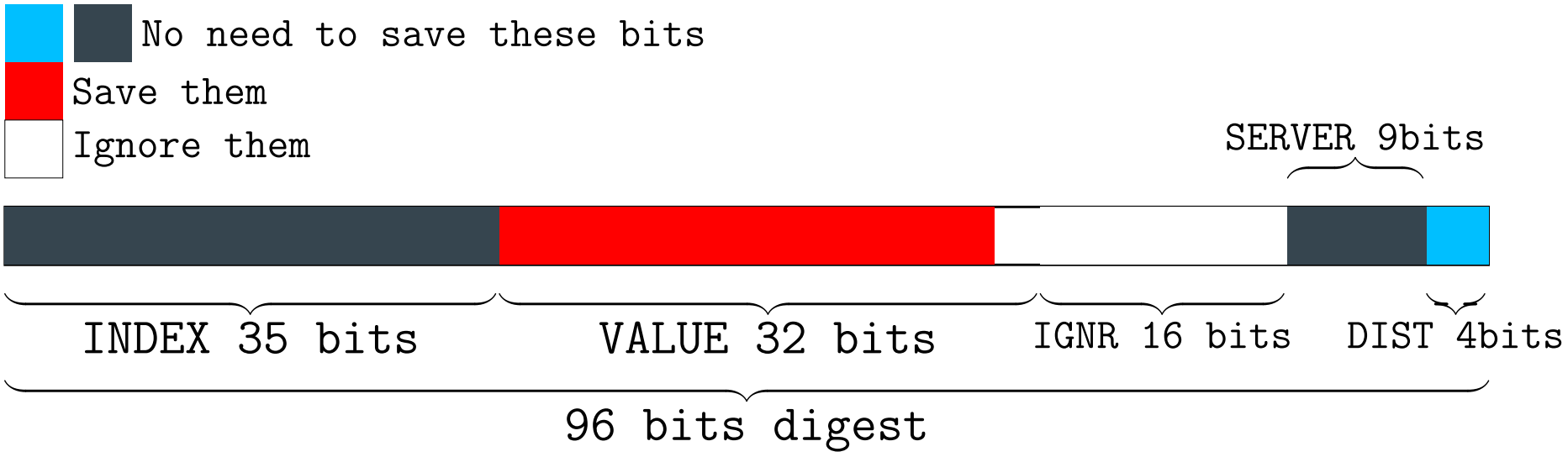
Linear Probing



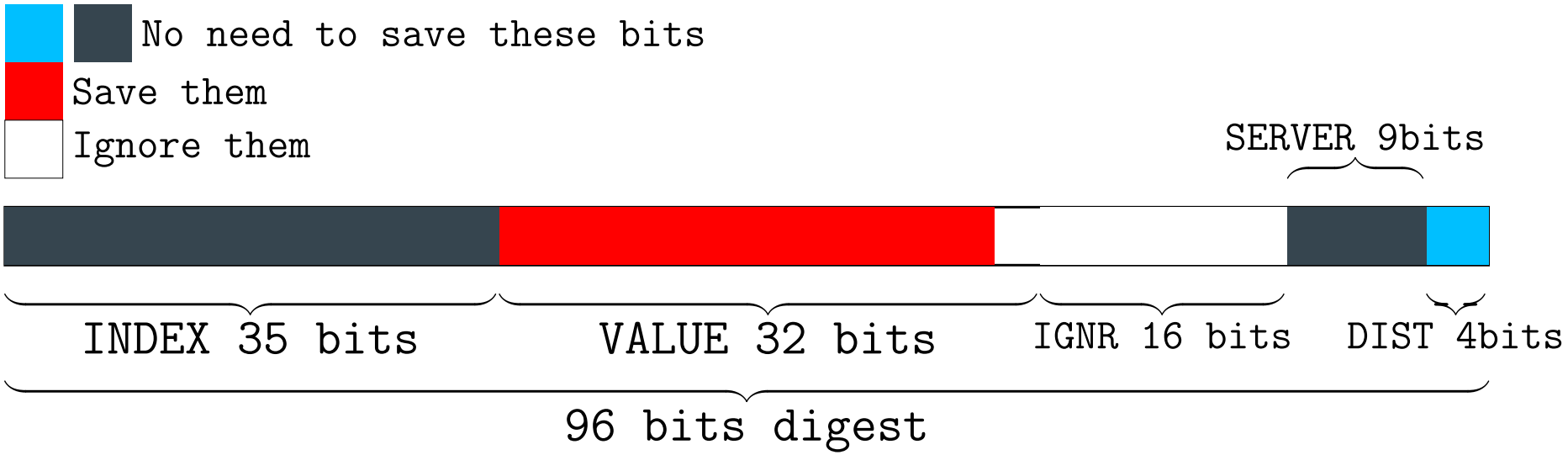
Linear Probing



Linear Probing

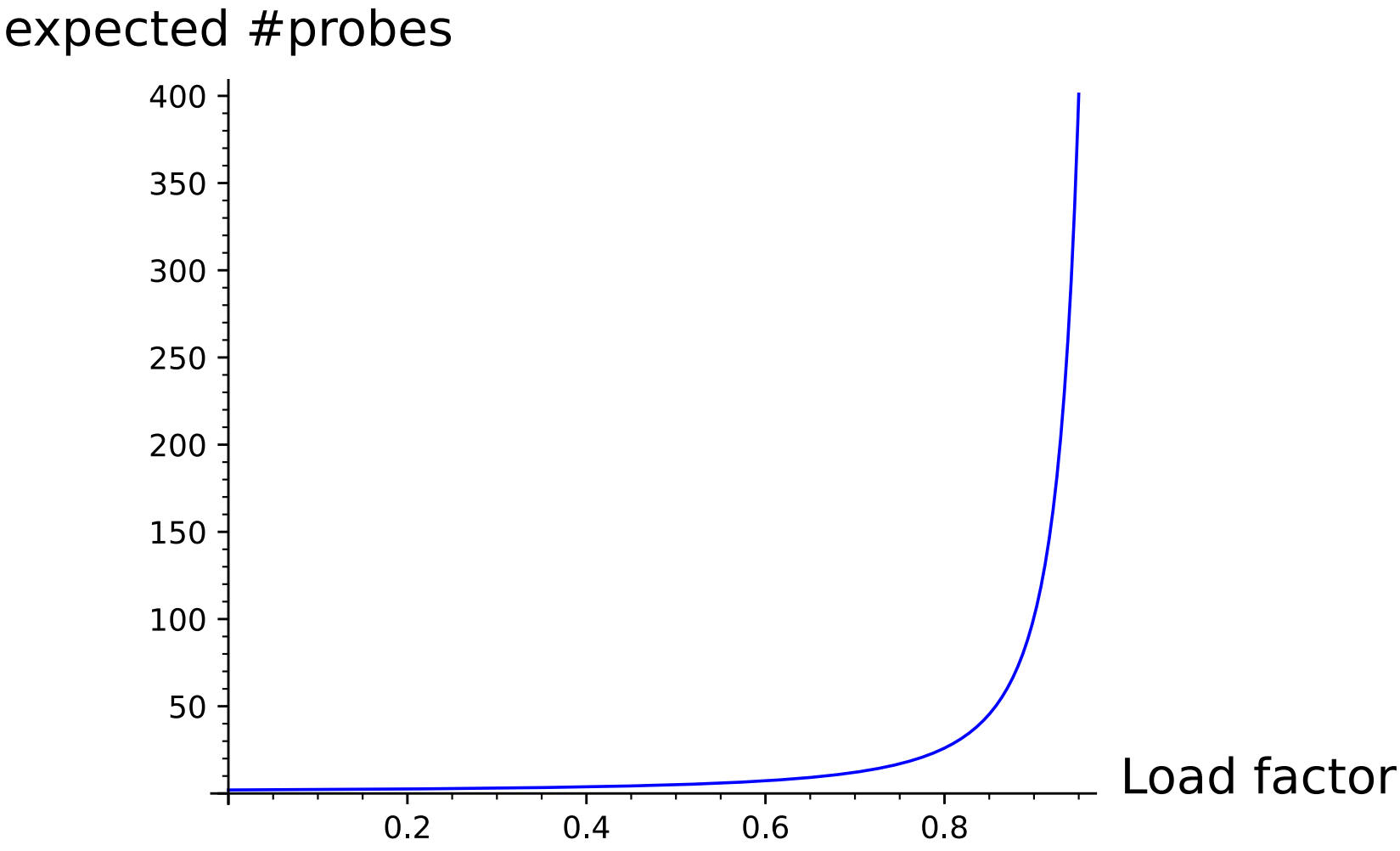


Linear Probing



- A Uses 3*x* less memory.
- small overhead from considering false positives.

Linear Probing



Experimenting with #probes

#Probes needed	% of elements
0	0%

Experimenting with #probes

#Probes needed	% of elements
0	0%
1	55%

Experimenting with #probes

#Probes needed	% of elements
0	0%
1	55%
2	69%

Experimenting with #probes

#Probes needed	% of elements
0	0%
1	55%
2	69%
3	75%

Experimenting with #probes

#Probes needed	% of elements
0	0%
1	55%
2	69%
3	75%
4	76%

Experimenting with #probes

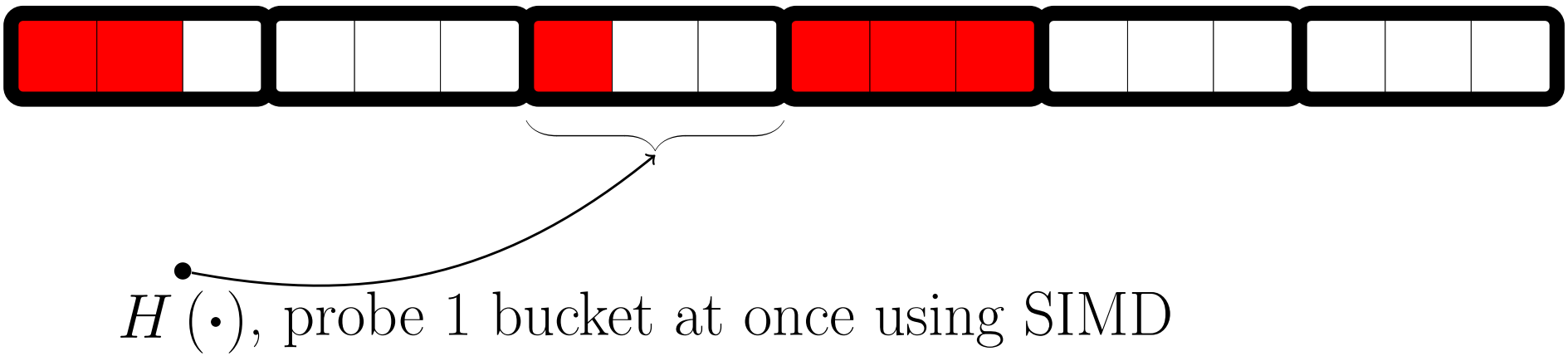
#Probes needed	% of elements
0	0%
1	55%
2	69%
3	75%
4	76%
6	80%
7	83%
8	85%
9	87%
10	88%
11	89%
12	90%
13	90.9%
14	91%
15	92%

Experimenting with #probes

#Probes needed	% of elements
0	0%
1	55%
2	69%
3	75%
4	76%
6	80%
7	83%
10	88%
11	89%
12	90%
13	90.9%
14	91%
15	92%

Discarding 10% of annoying elements enables constant time probing and insertion, with only 10% of false negatives.

Linear Probing



- Instead of 4,660 cpu hours, we need 290,816 cpu hours.
- The attack cost jumped from 140€to 8724€.
- Using only disitinguished point the attack would cost:
  - 1,638,400 cpu hours
  - i.e. 49k €

Thank you!