Over the Edge:

Silently Owning Windows 10's Secure Browser

VU VRIJE UNIVERSITEIT AMSTERDAM

Erik Bosman, Kaveh Razavi, Herbert Bos and Cristiano Giuffrida

JULY 30 - AUGUST 4, 2016 / MANDALAY BAY / LAS VEGAS

WARNING

THIS PRESENTATION
MAY CONTAIN POINTERS

This presentation:

Deduplication
(software side-channel)

This presentation:

Deduplication
(software side-channel)
+
Rowhammer
(hardware bug)

This presentation:

Deduplication
(software side-channel)
+
Rowhammer
(hardware bug)

Exploit MS Edge without software bugs
(from JavaScript)

This presentation:

Deduplication
(software side-char
+
Rowhammer
(hardware bug)

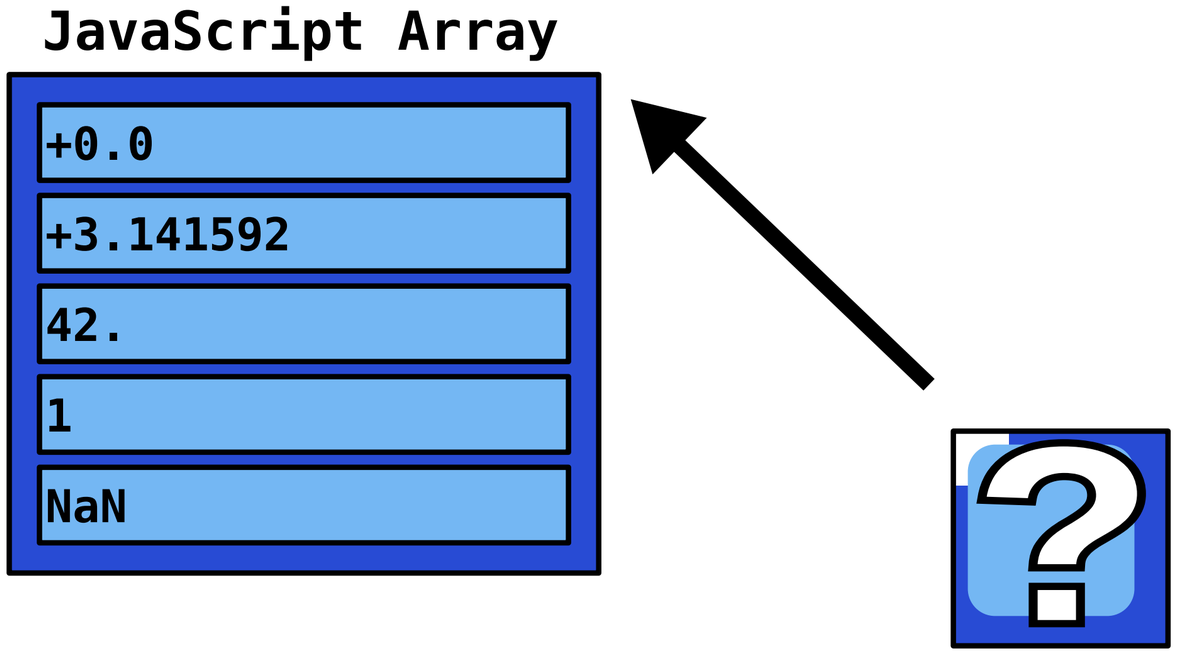
↓

Exploit MS Edge without software bugs
(from JavaScript)

Outline:

Deduplication

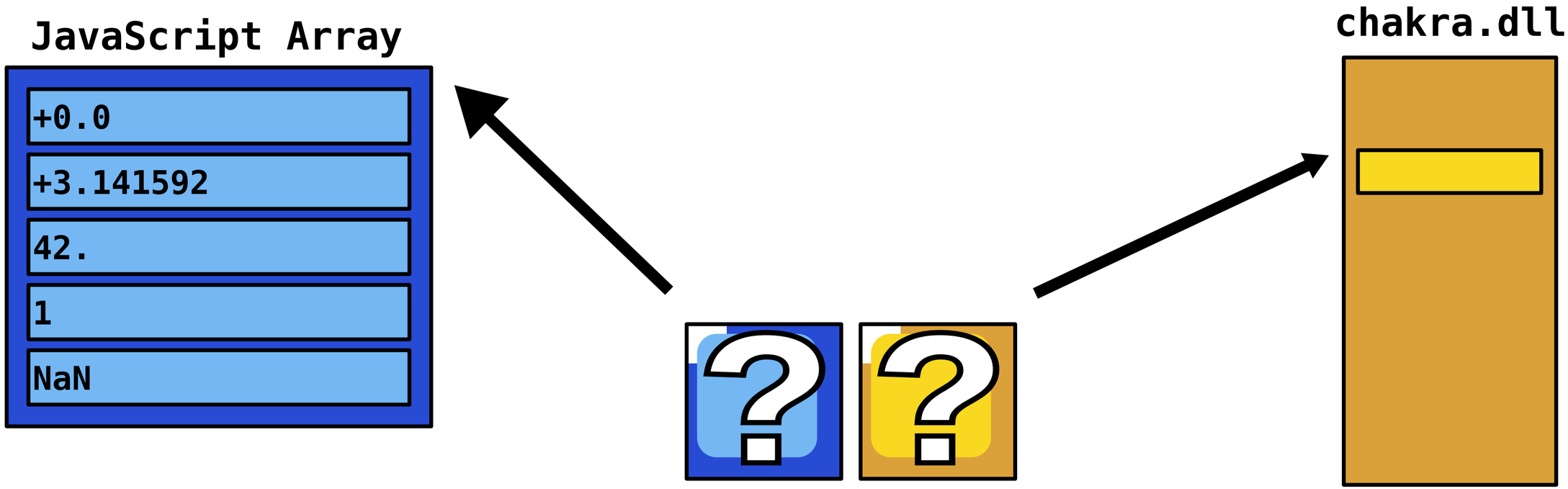
- leak heap & code addresses



Outline:

Deduplication

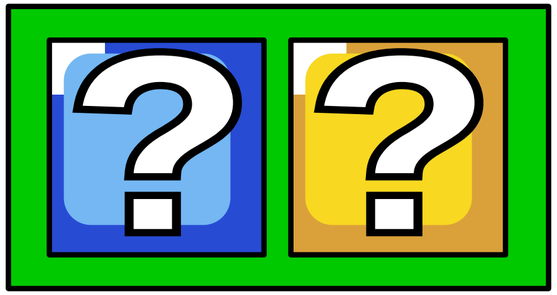
- leak heap & code addresses



Outline:

Deduplication

- leak heap & code addresses
- create a fake object



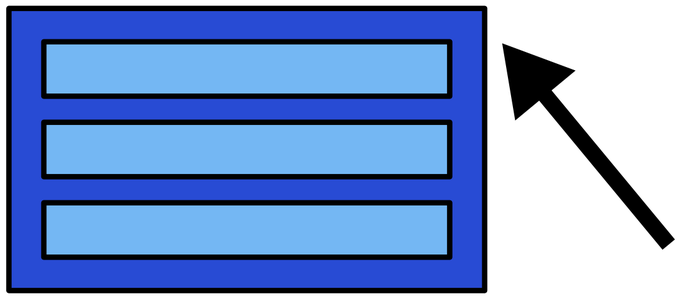
Outline:

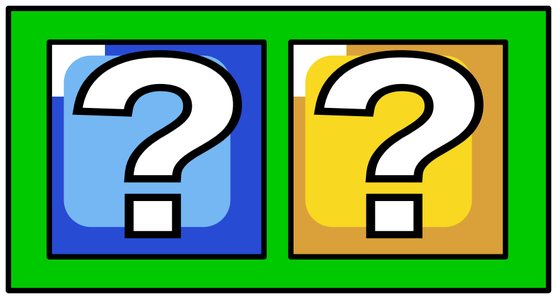
Deduplication

- leak heap & code addresses
- create a fake object

Rowhammer

- create reference to our fake object





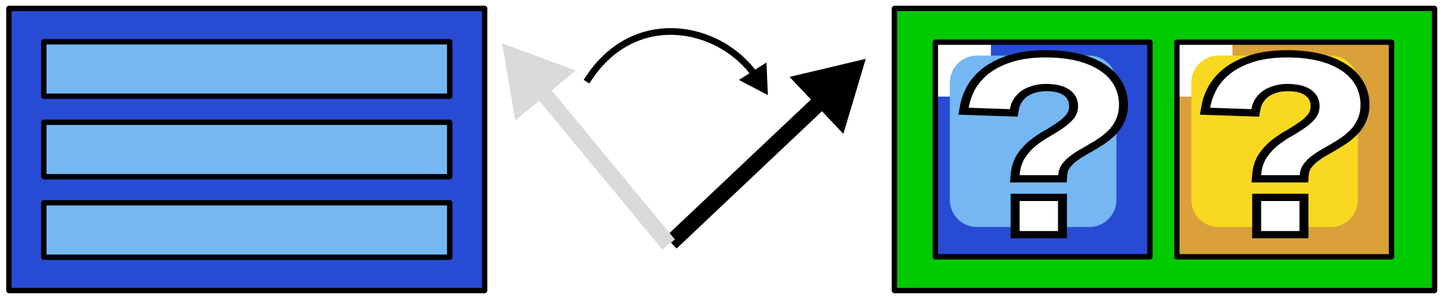
Outline:

Deduplication

- leak heap & code addresses
- create a fake object

Rowhammer

- create reference to our fake object



memory deduplication

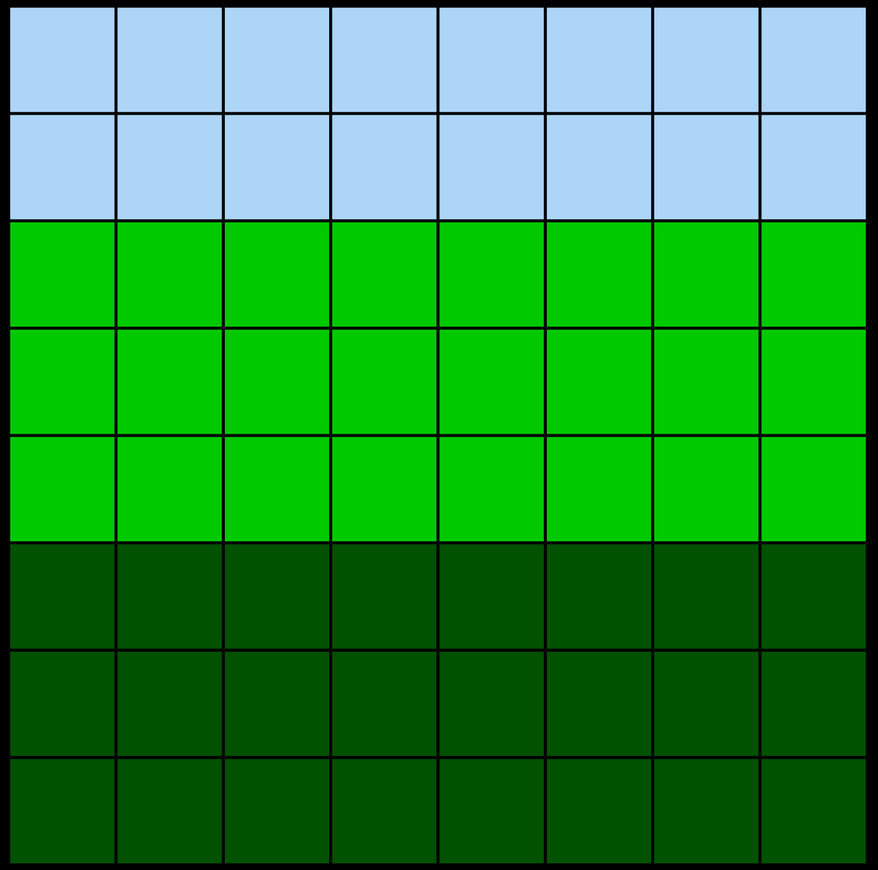
A method of reducing memory usage.

Used in virtualisation environments,

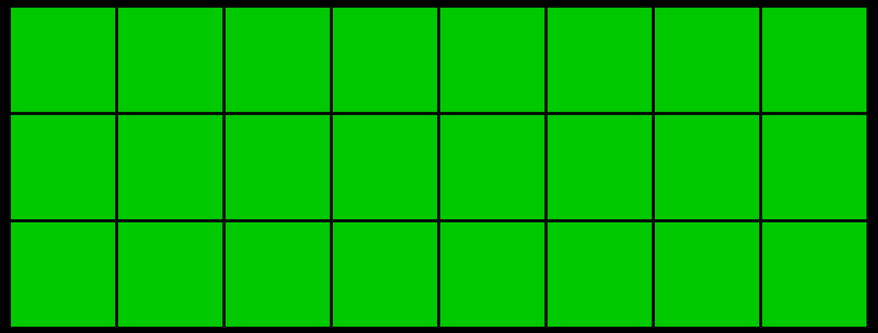
(was) also enabled by default on Windows 8.1 and 10.

memory deduplication

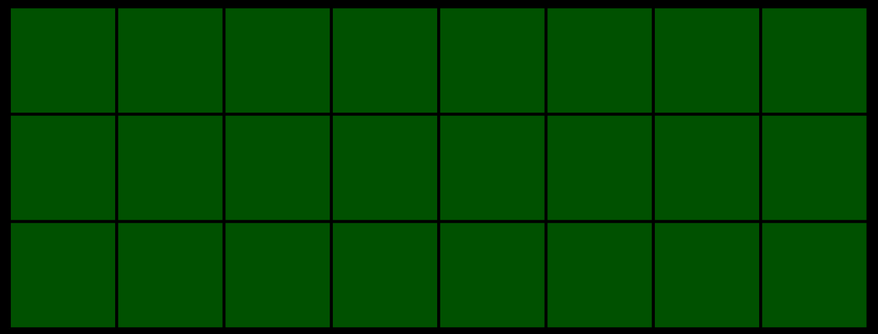
physical memory



process A

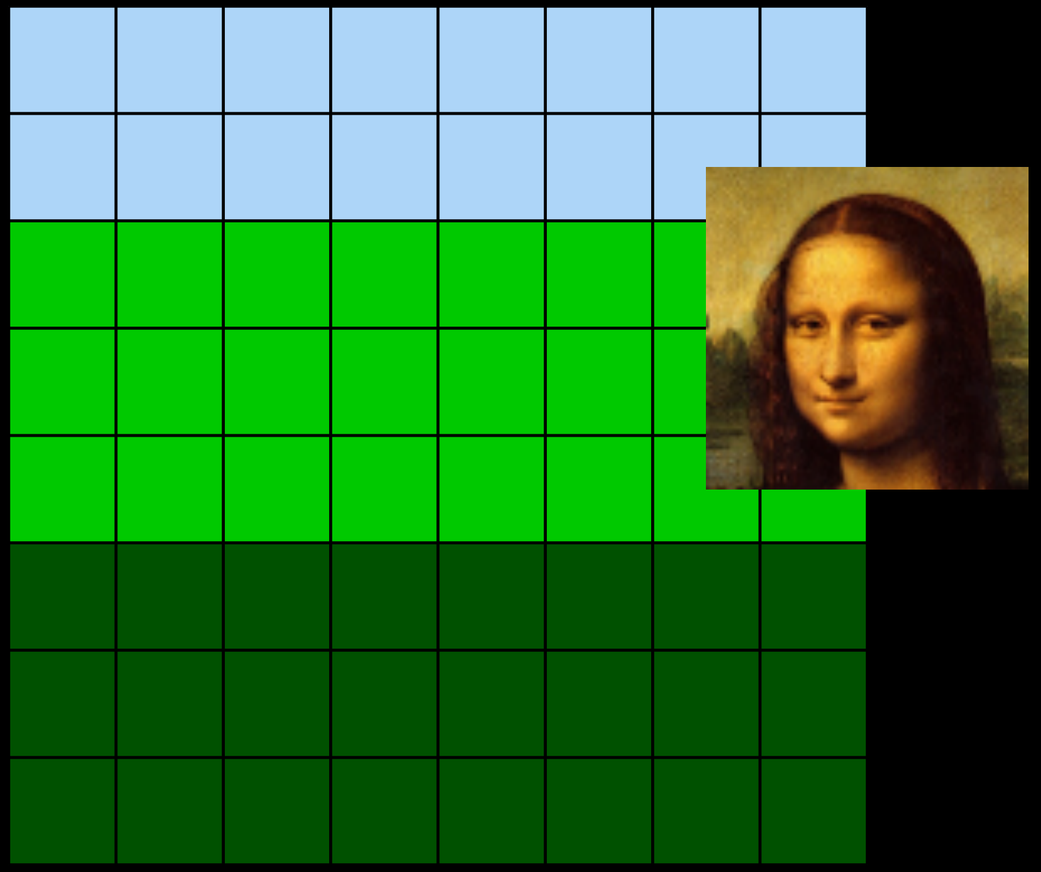


process B

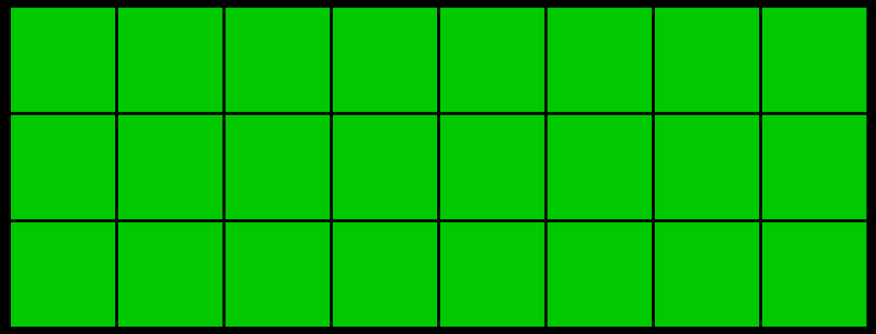


memory deduplication

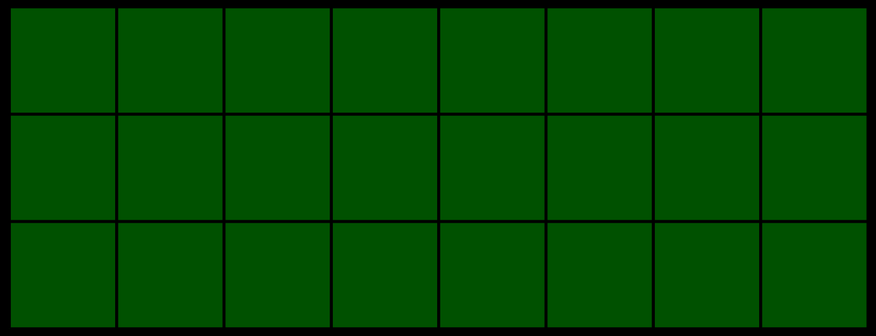
physical memory



process A

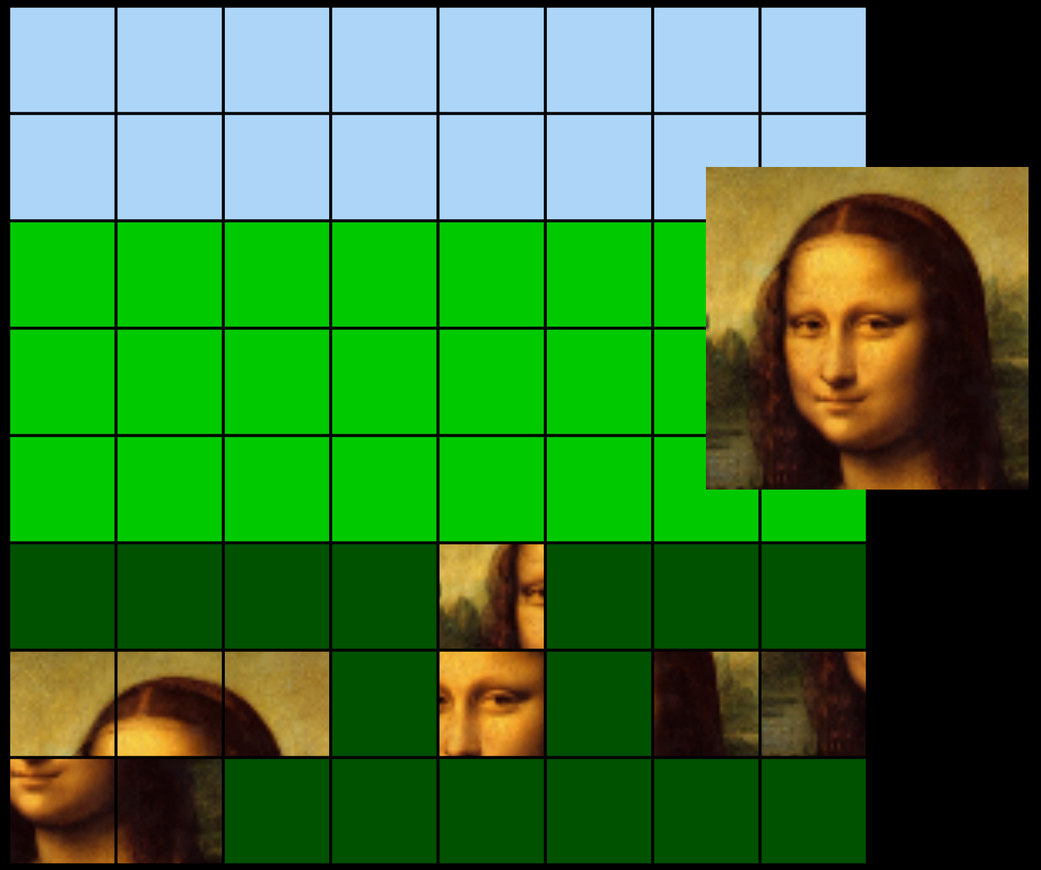


process B

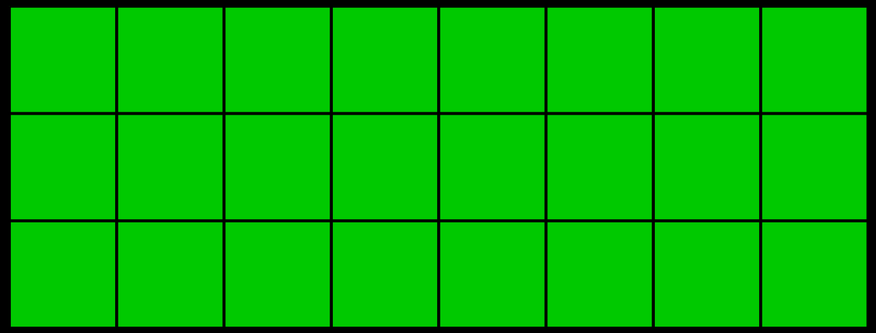


memory deduplication

physical memory



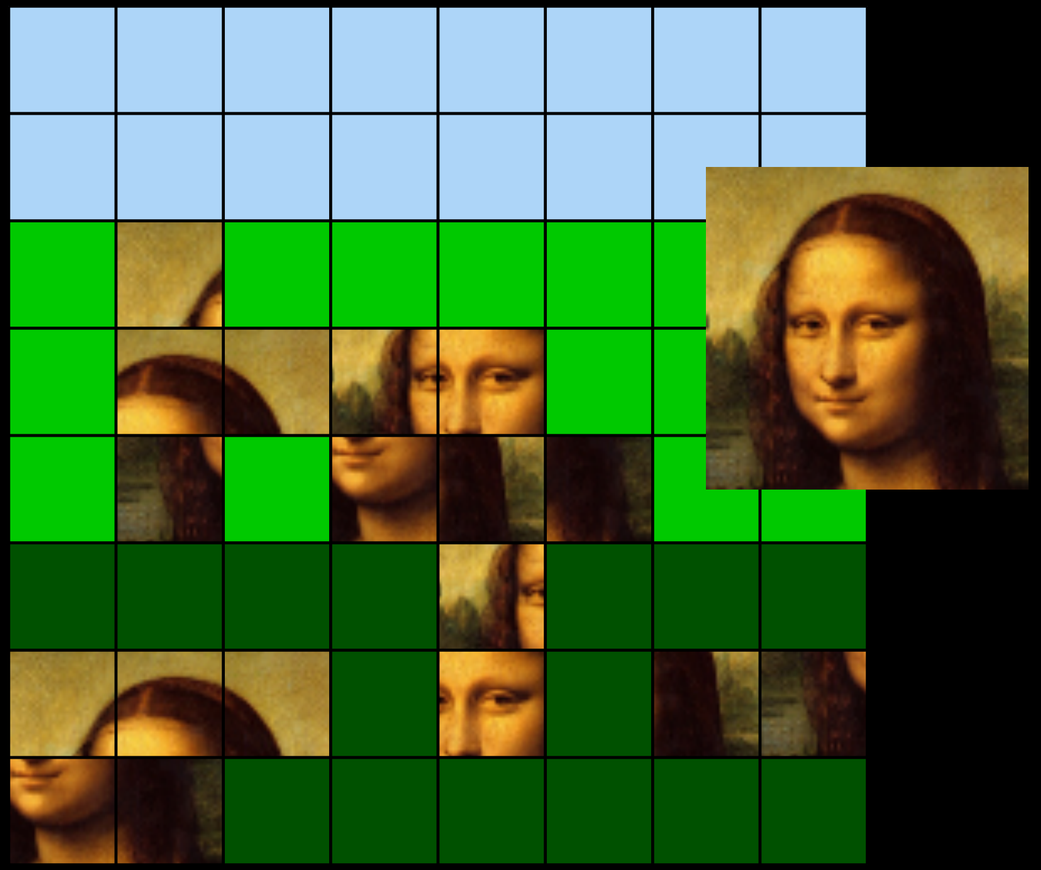
process A



process B

memory deduplication

physical memory

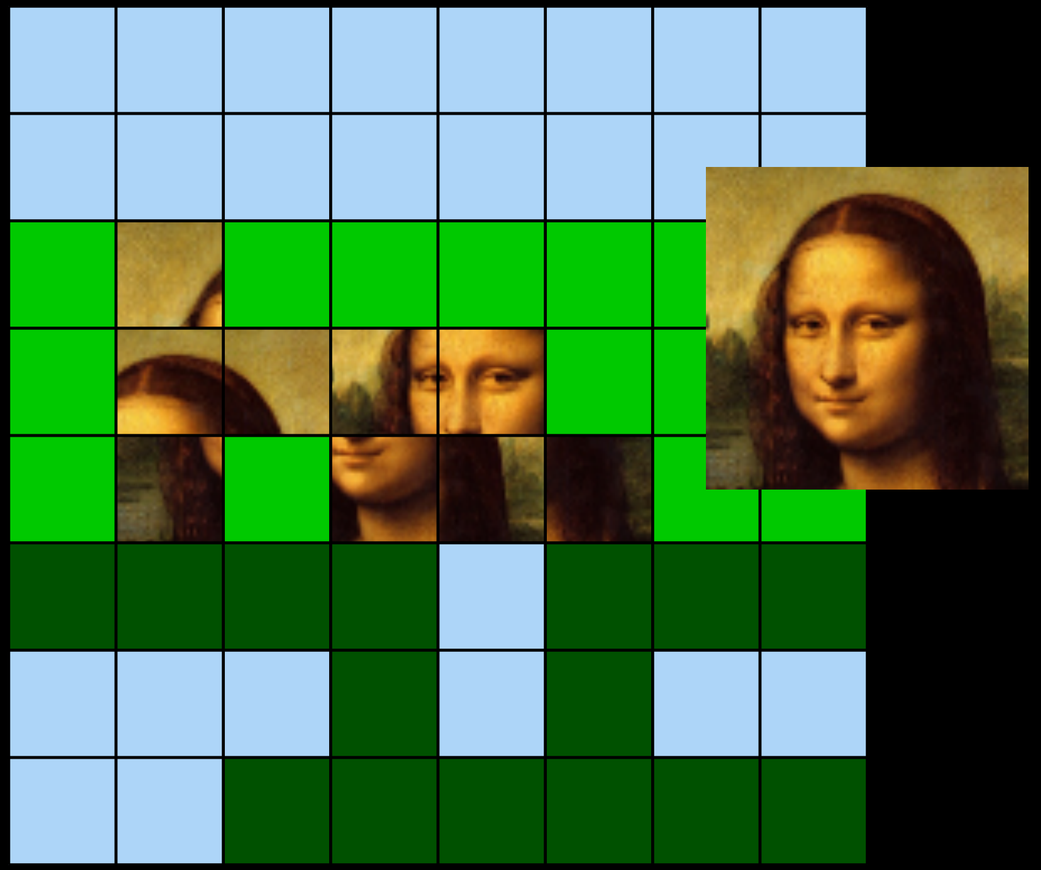


process A

process B

memory deduplication

physical memory

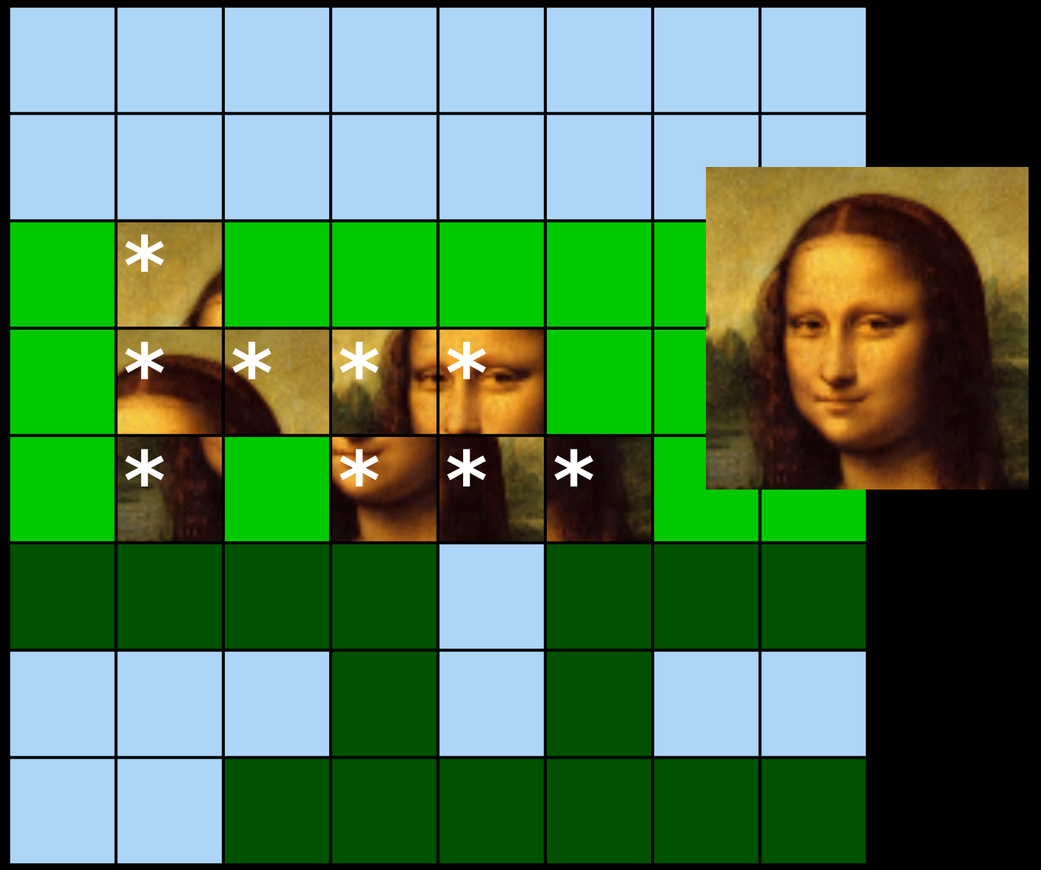


process A

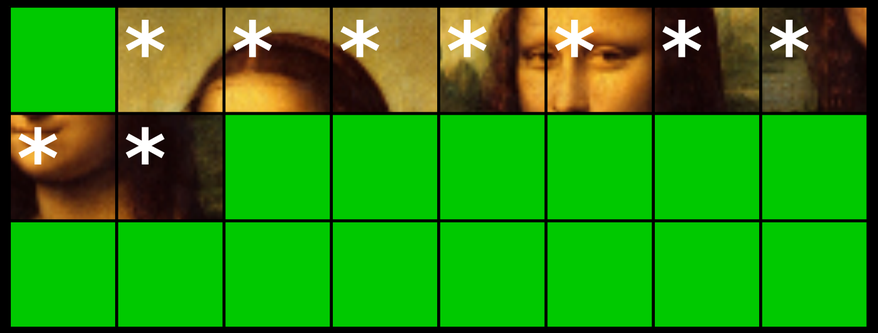
process B

memory deduplication

physical memory



process A



process B

memory deduplication: The Problem

Deduplicated memory does not need to have the same origin.

(unlike fork(), file-backed memory)

An attacker can use deduplication as a side-channel

deduplication side-channel attack

normal write

deduplication side-channel attack

normal write

deduplication side-channel attack

normal write

copy on write (due to deduplication)

deduplication side-channel attack

normal write

copy on write (due to deduplication)

deduplication side-channel attack

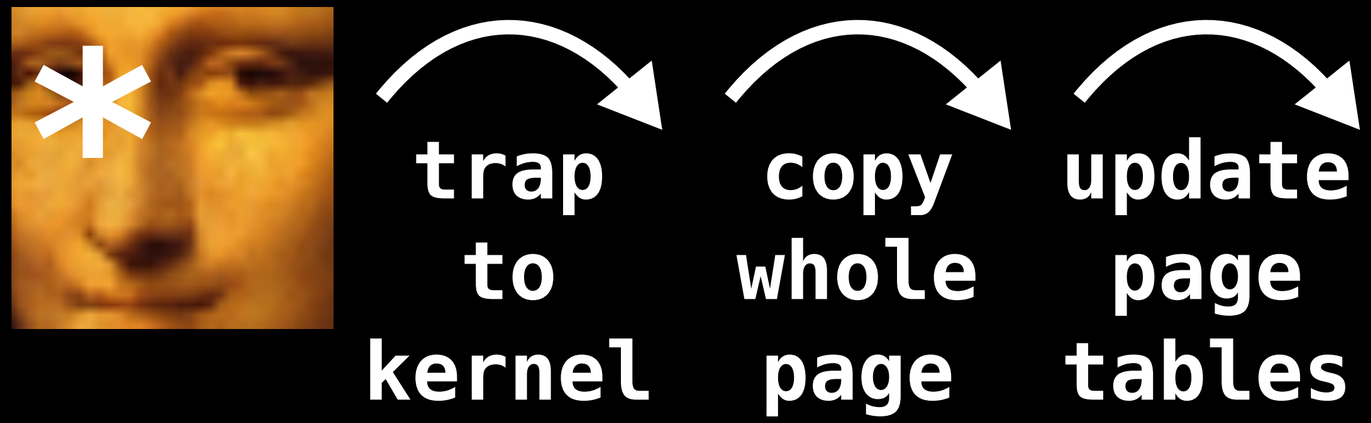
normal write

copy on write (due to deduplication)

deduplication side-channel attack

normal write

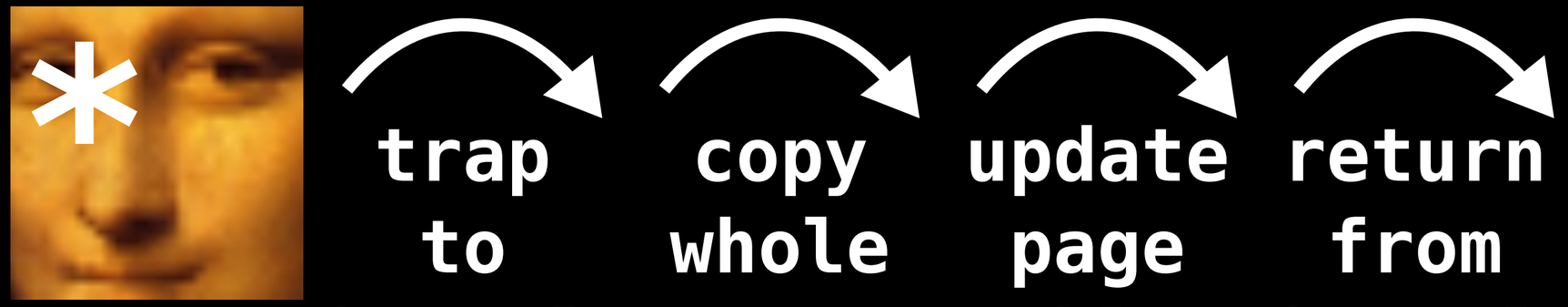
copy on write (due to deduplication)



deduplication side-channel attack

normal write

copy on write (due to deduplication)



deduplication side-channel attack

normal write

copy on write (due to deduplication)

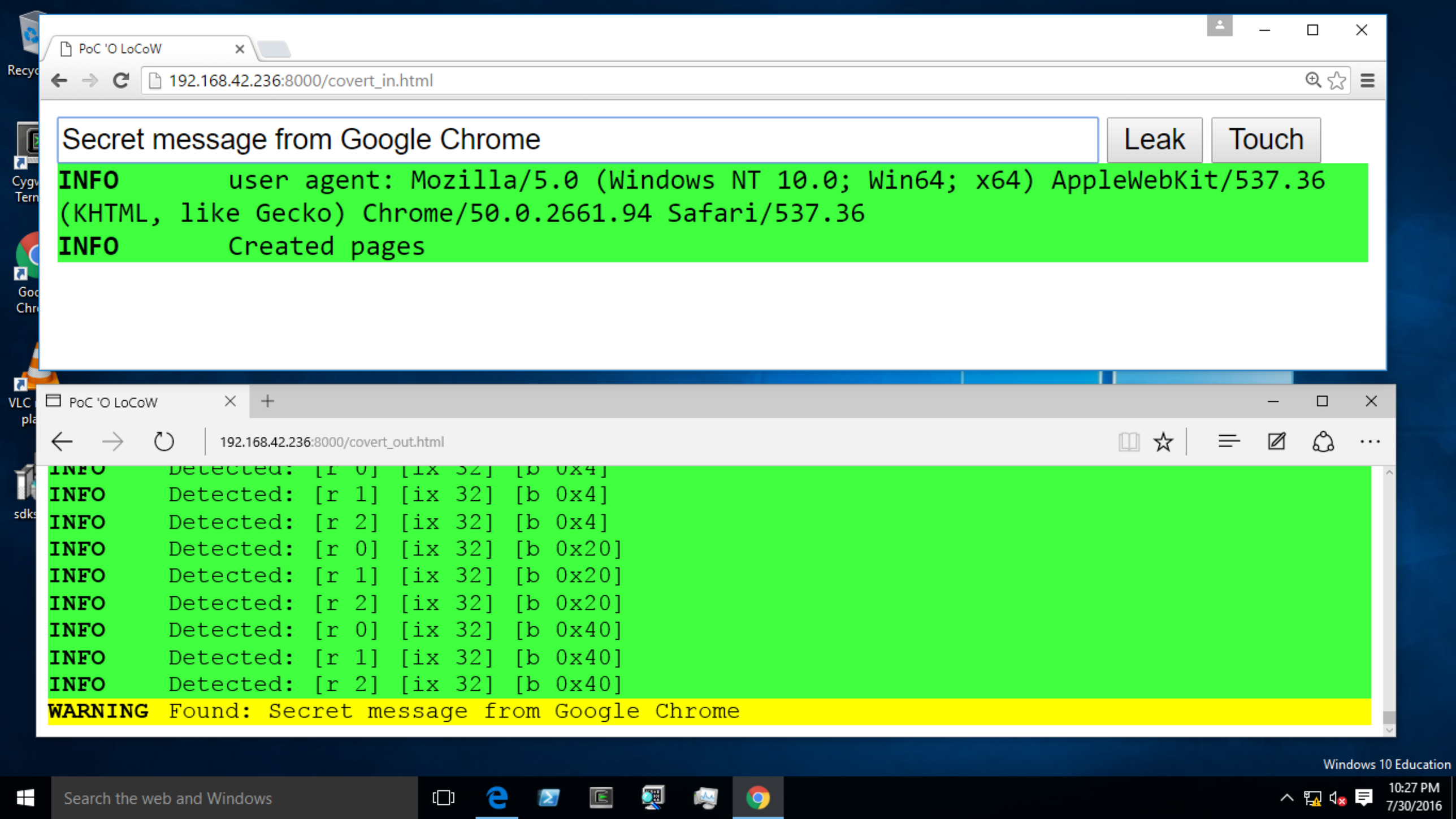
deduplication side-channel attack

A 1-bit side channel which is able
to leak data across security boundaries

- cross VM
- cross-process
- leak process data from javascript code

having fun with deduplication

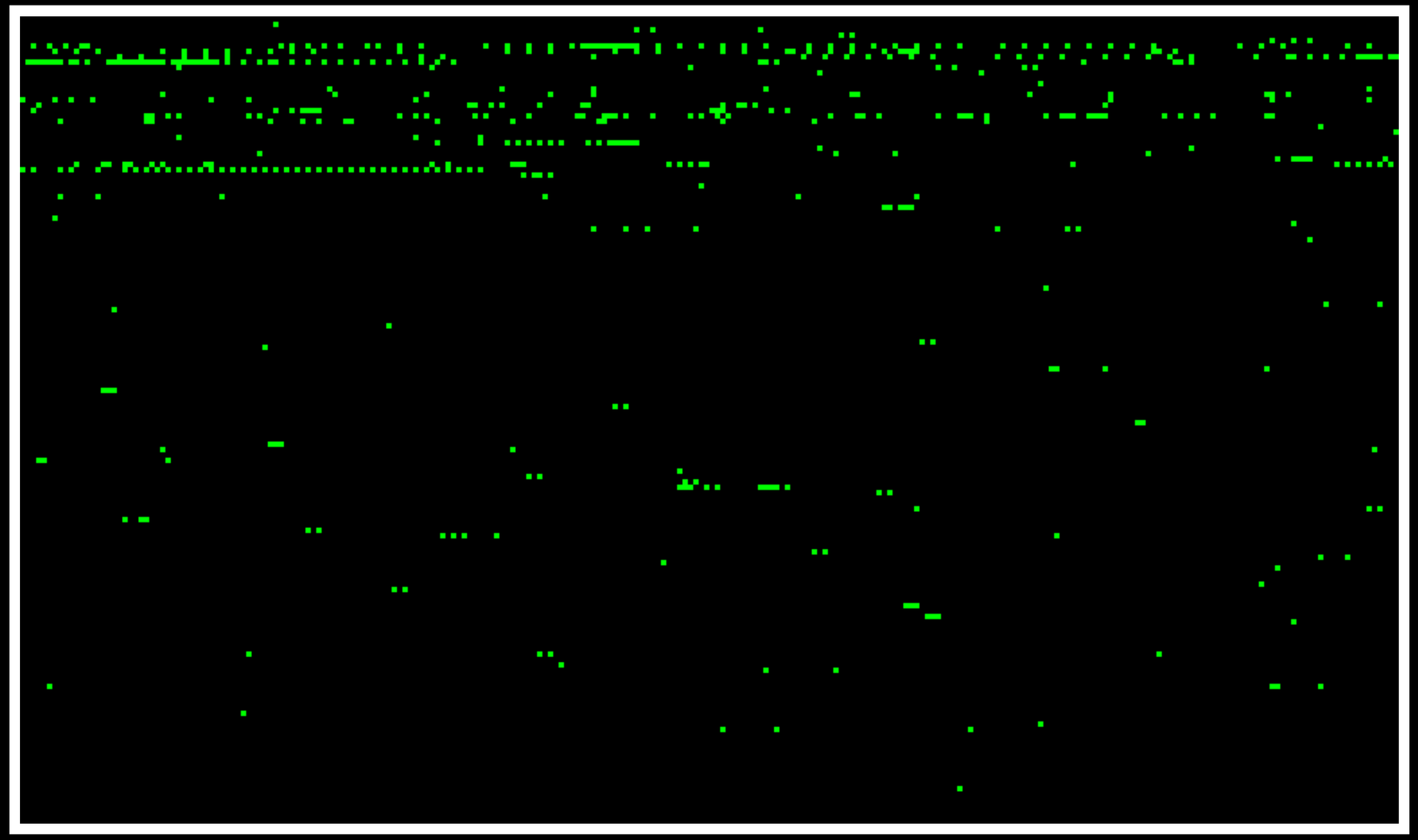
- covert channel


PoC 'O LoCoW
192.168.42.236:8000/covert_in.html
Secret message from Google Chrome
Leak
Touch
INFO user agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.94 Safari/537.36
INFO Created pages
PoC 'O LoCoW
192.168.42.236:8000/covert_out.html
INFO Detected: [r 1] [ix 32] [b 0x4]
INFO Detected: [r 2] [ix 32] [b 0x4]
INFO Detected: [r 0] [ix 32] [b 0x20]
INFO Detected: [r 1] [ix 32] [b 0x20]
INFO Detected: [r 2] [ix 32] [b 0x20]
INFO Detected: [r 0] [ix 32] [b 0x40]
INFO Detected: [r 1] [ix 32] [b 0x40]
INFO Detected: [r 2] [ix 32] [b 0x40]
WARNING Found: Secret message from Google Chrome
Windows 10 Education
Search the web and Windows
10:27 PM
7/30/2016

having fun with deduplication

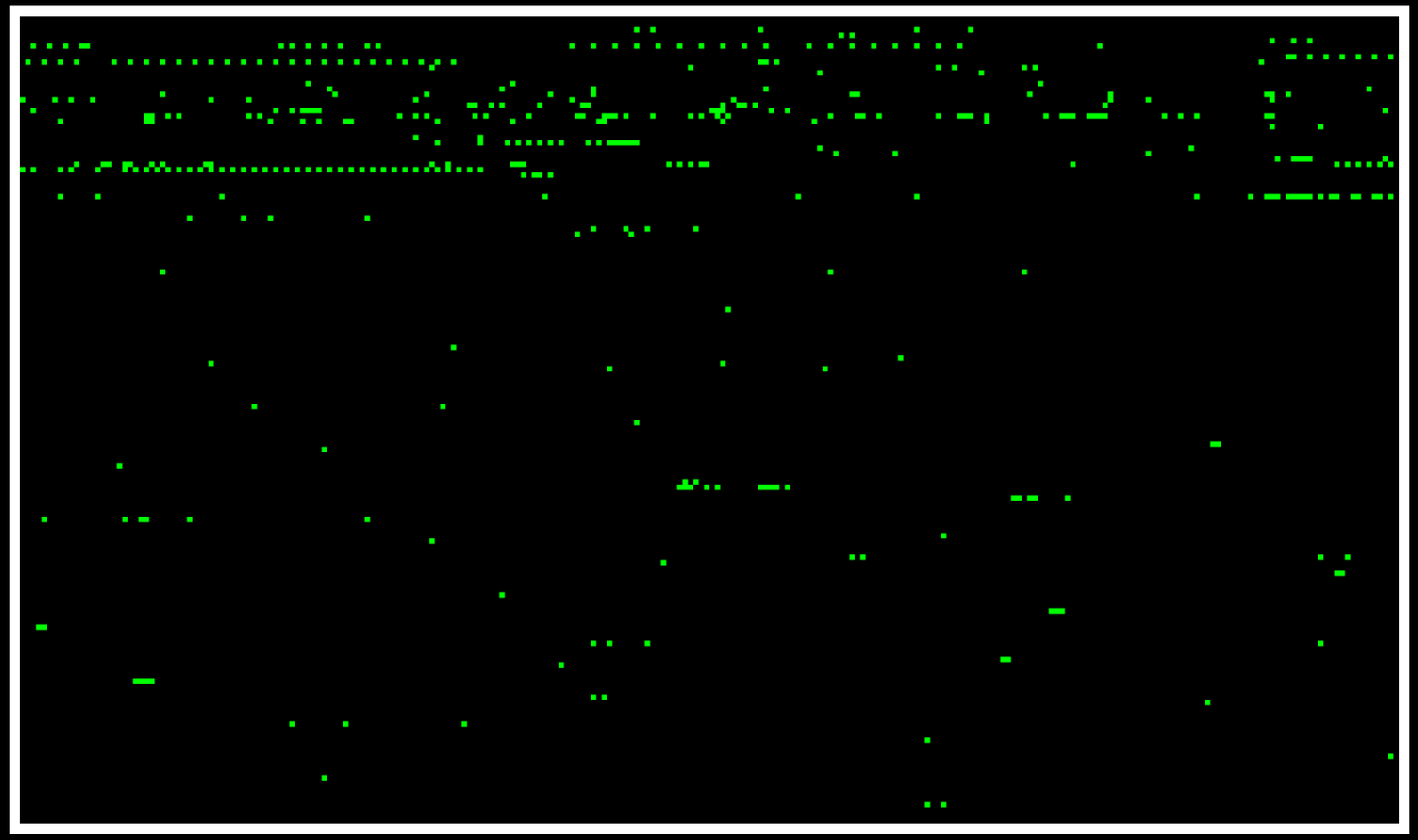
- covert channel
- detect running software

Wordpad memory dump



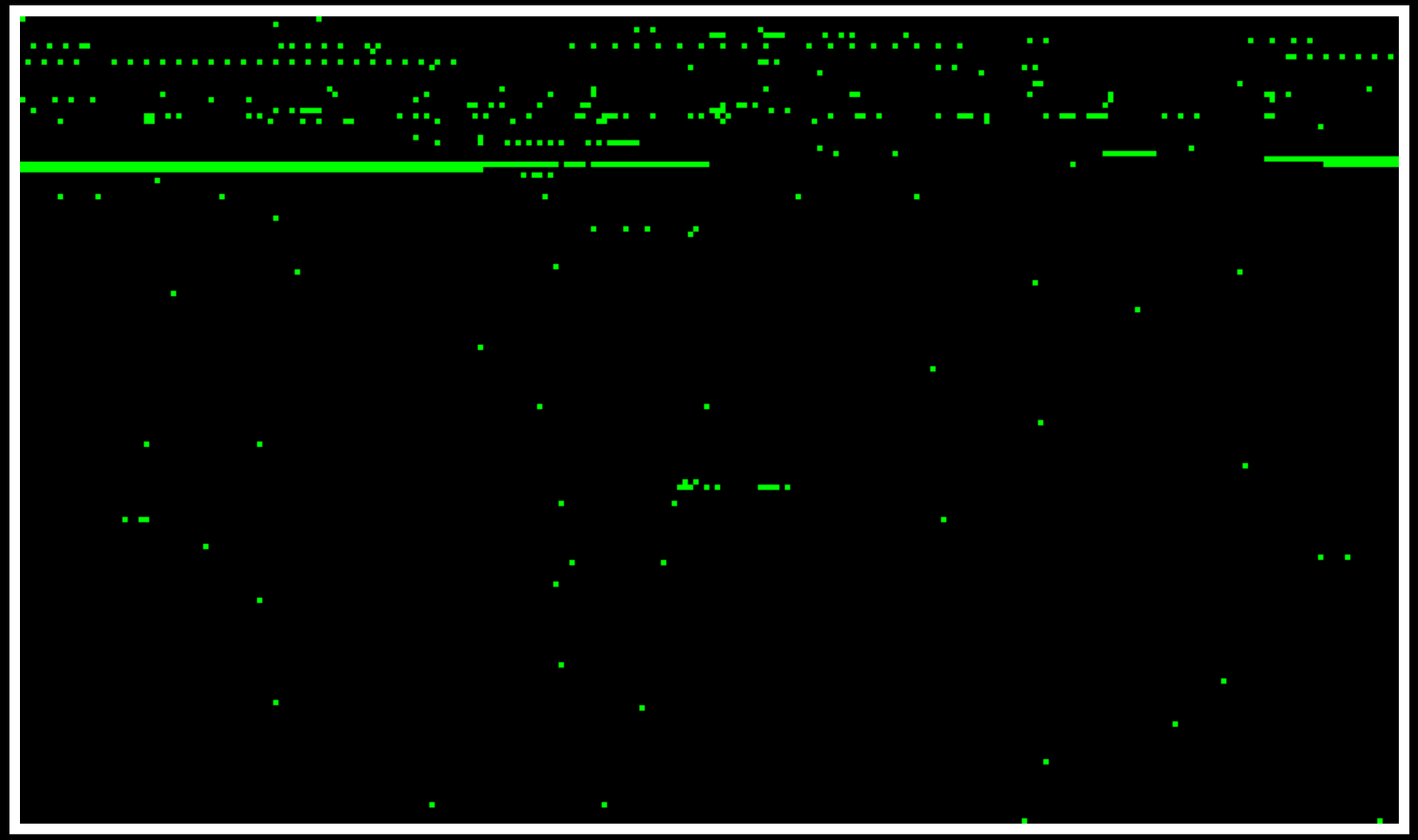
wordpad not running

Wordpad memory dump



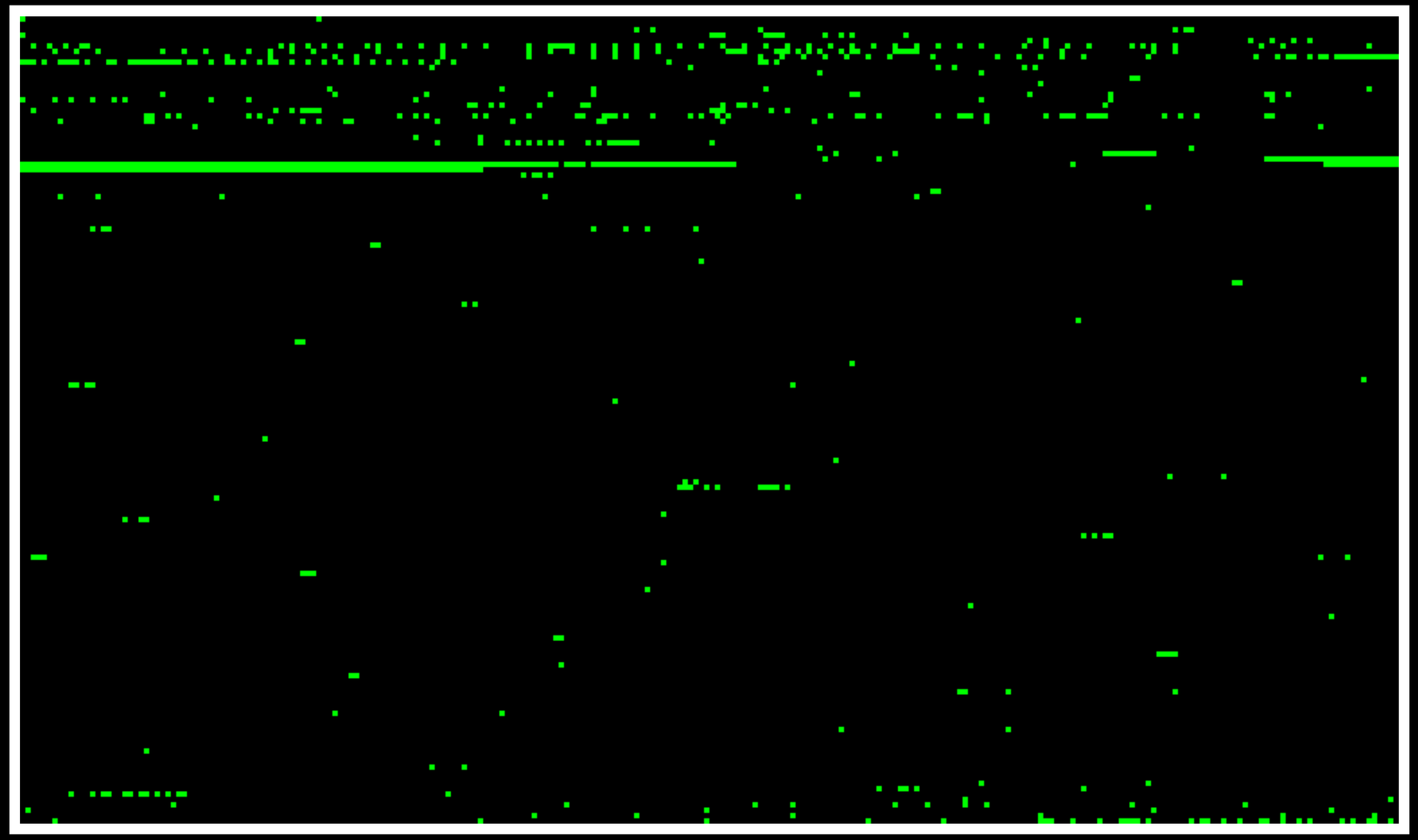
wordpad not running

Wordpad memory dump



wordpad running

Wordpad memory dump



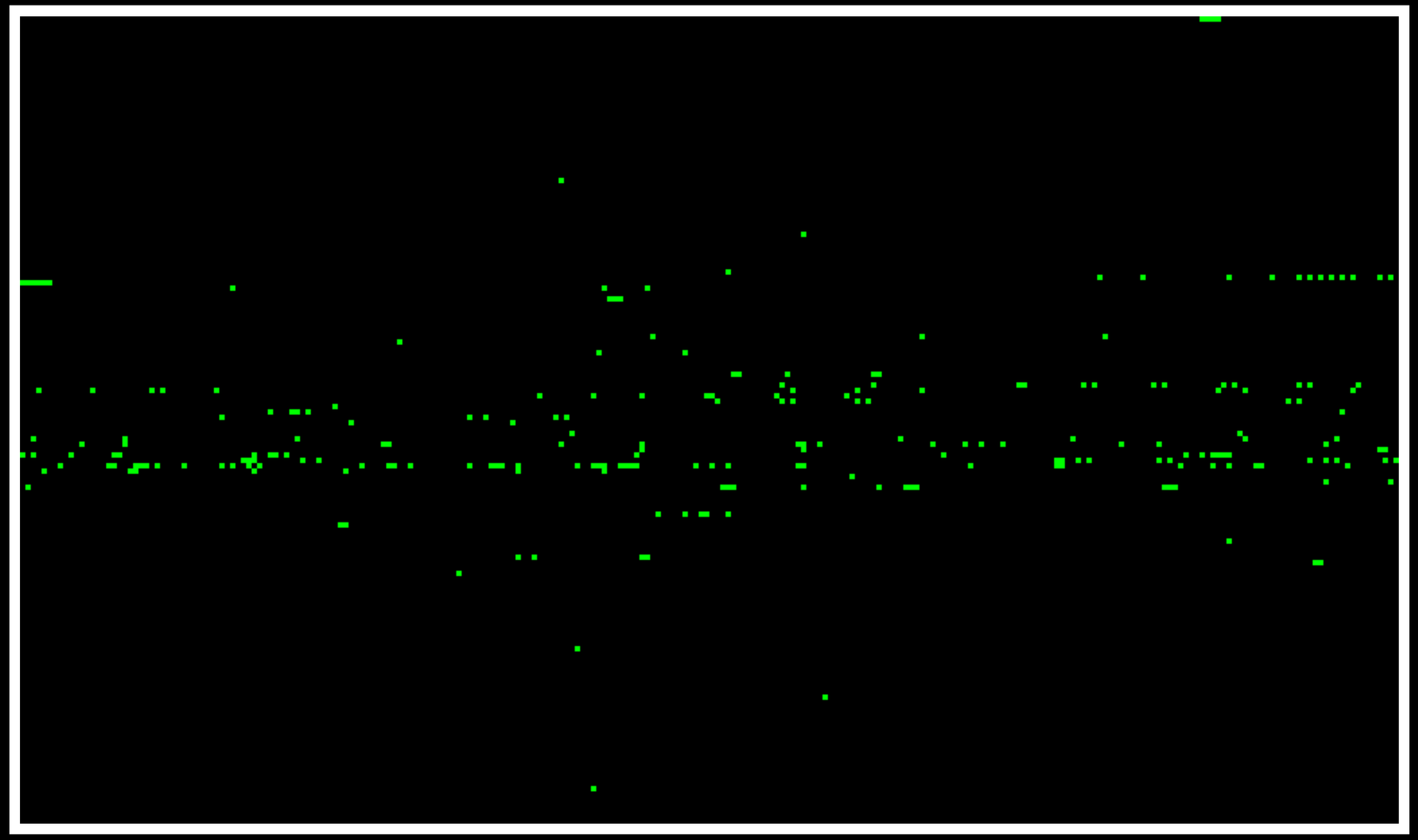
wordpad running

Signal not as clear as expected,

Reason:

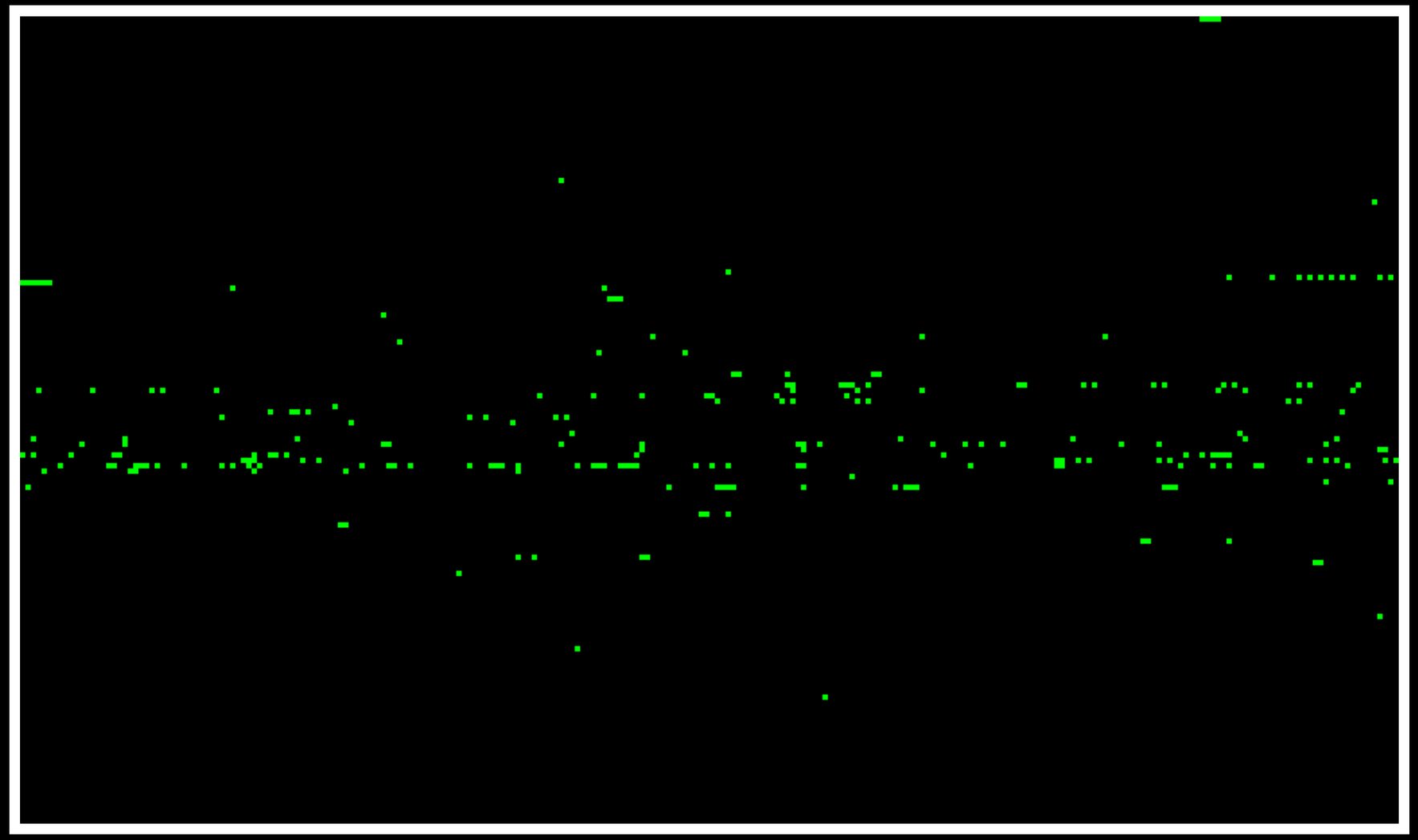
- file backed memory
- not deduplicated the same way
- on Windows.

Skype memory dump



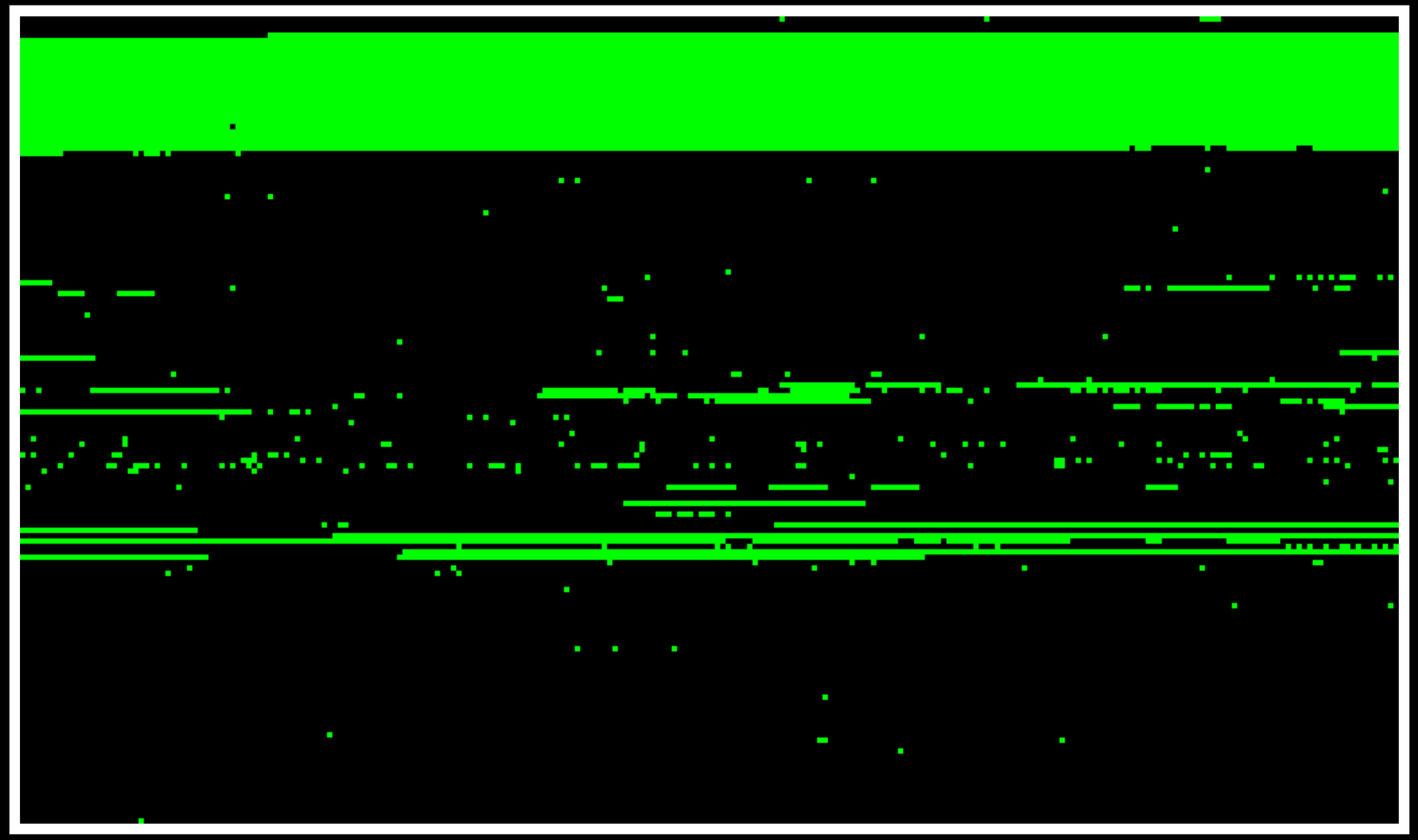
skype not running

Skype memory dump



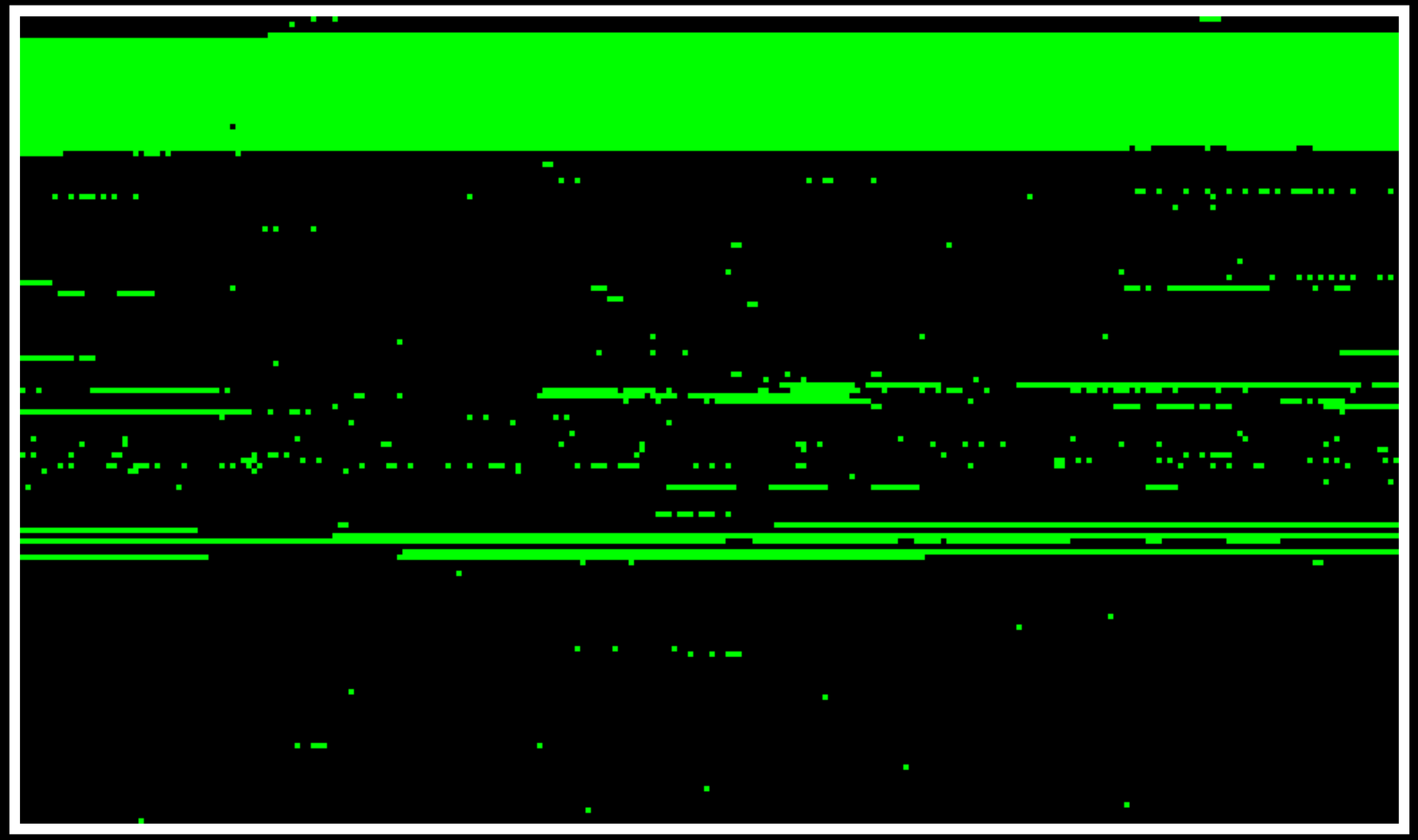
skype not running

Skype memory dump



skype running

Skype memory dump



skype running

For our Edge exploit,
a single-bit, page-granularity
info leak isn't enough

Can we generalize this to leaking arbitrary data, like an ASLR pointer or a password?

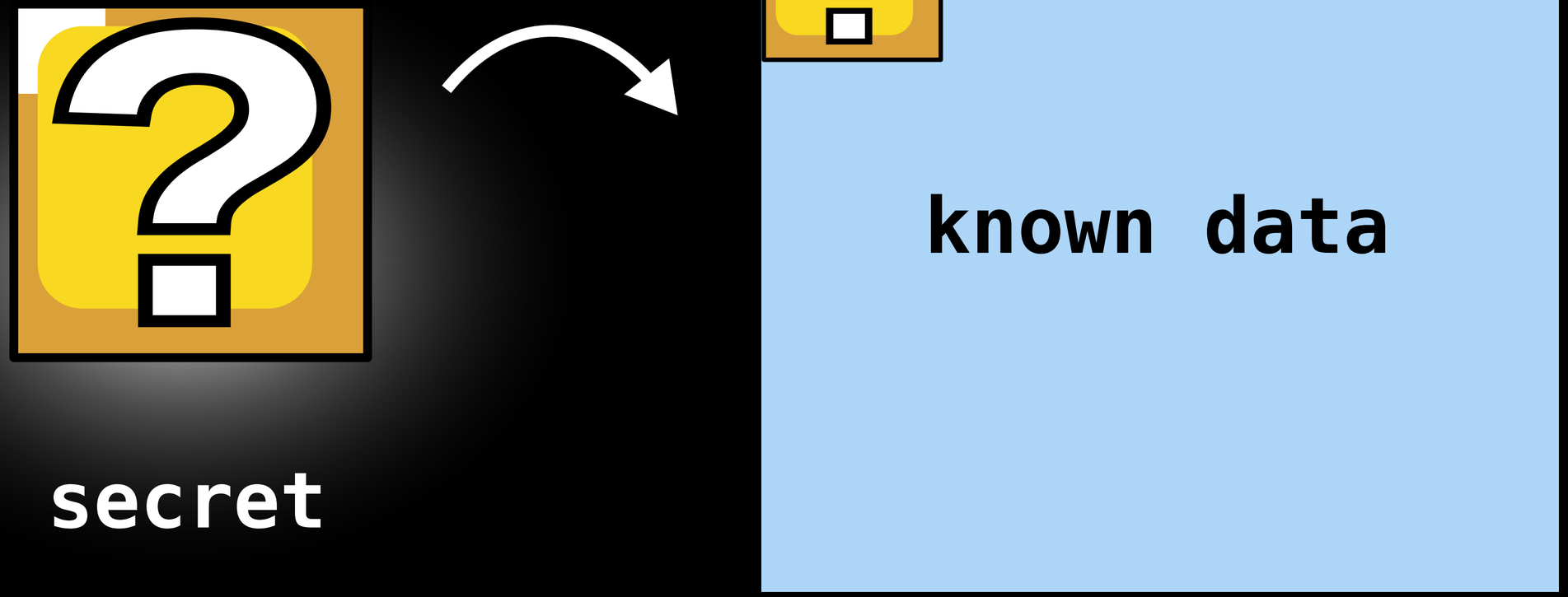
Challenge 1:

The secret we want to leak does not span an entire page.

turning a secret into a page

secret

turning a secret into a page



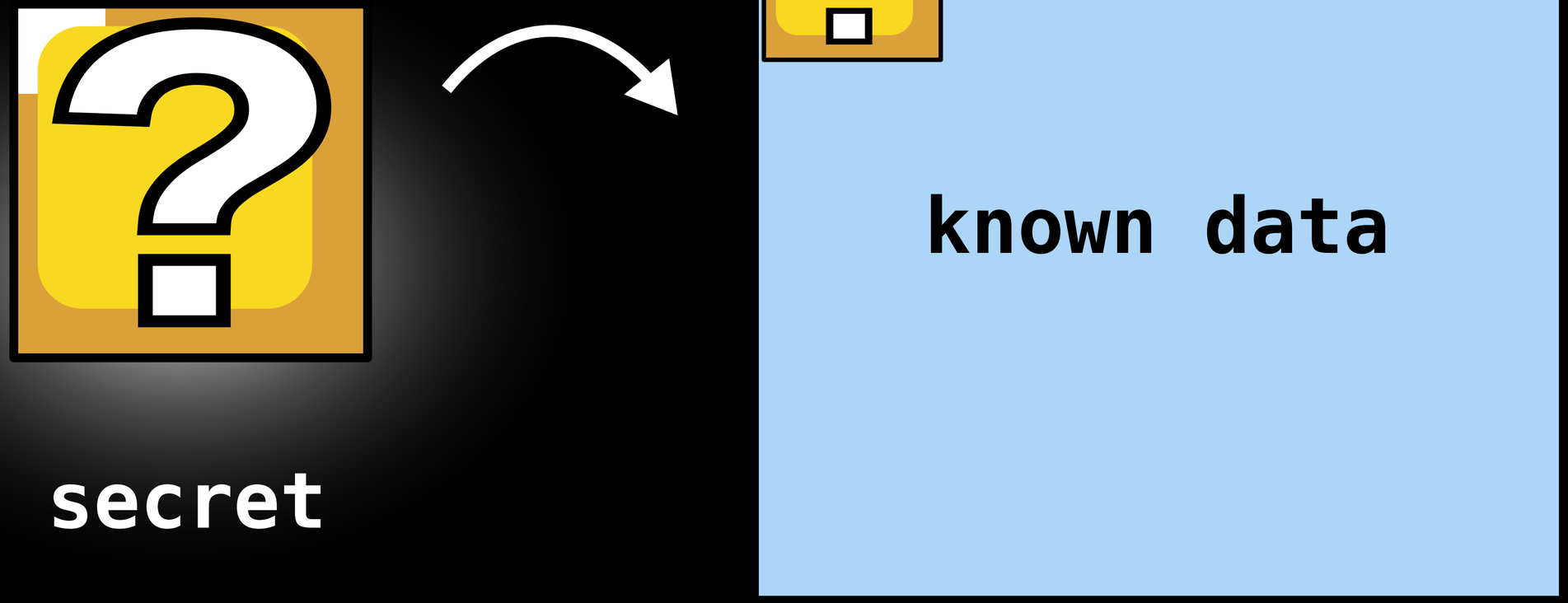
secret

secret page

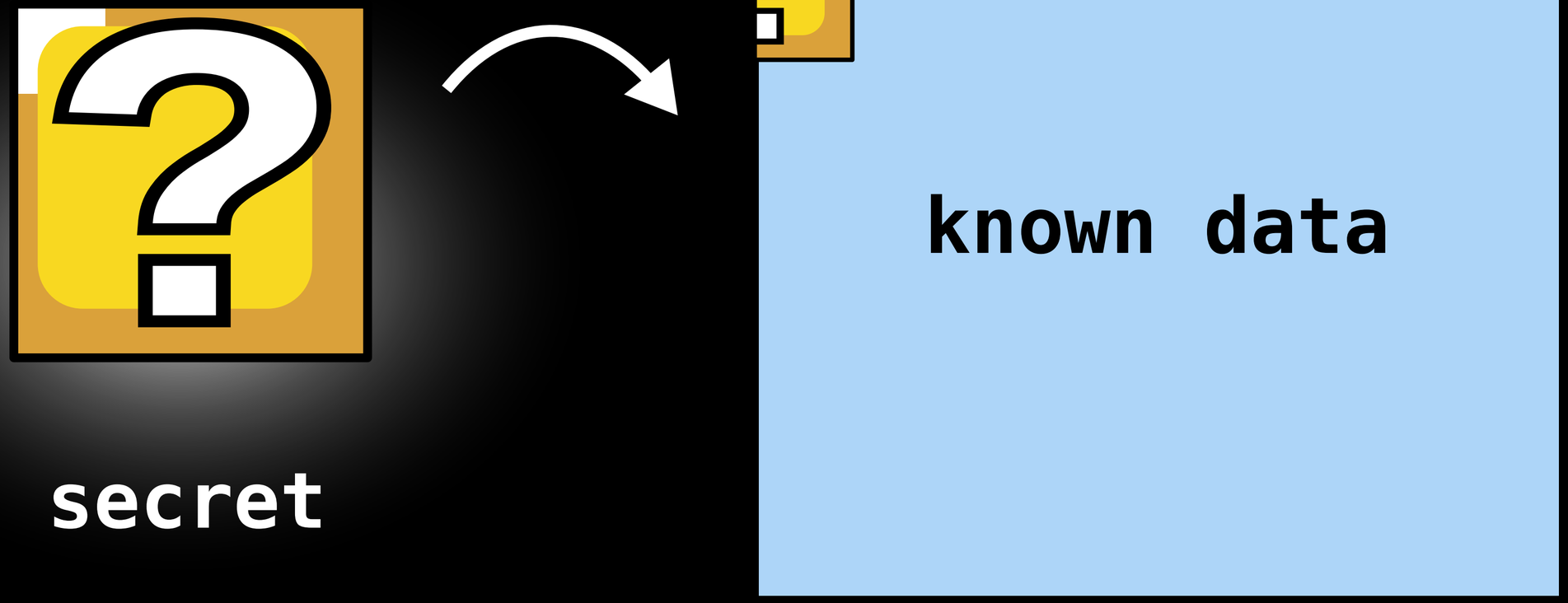
Challenge 2:

The secret we want to leak has too much entropy to leak all at once.

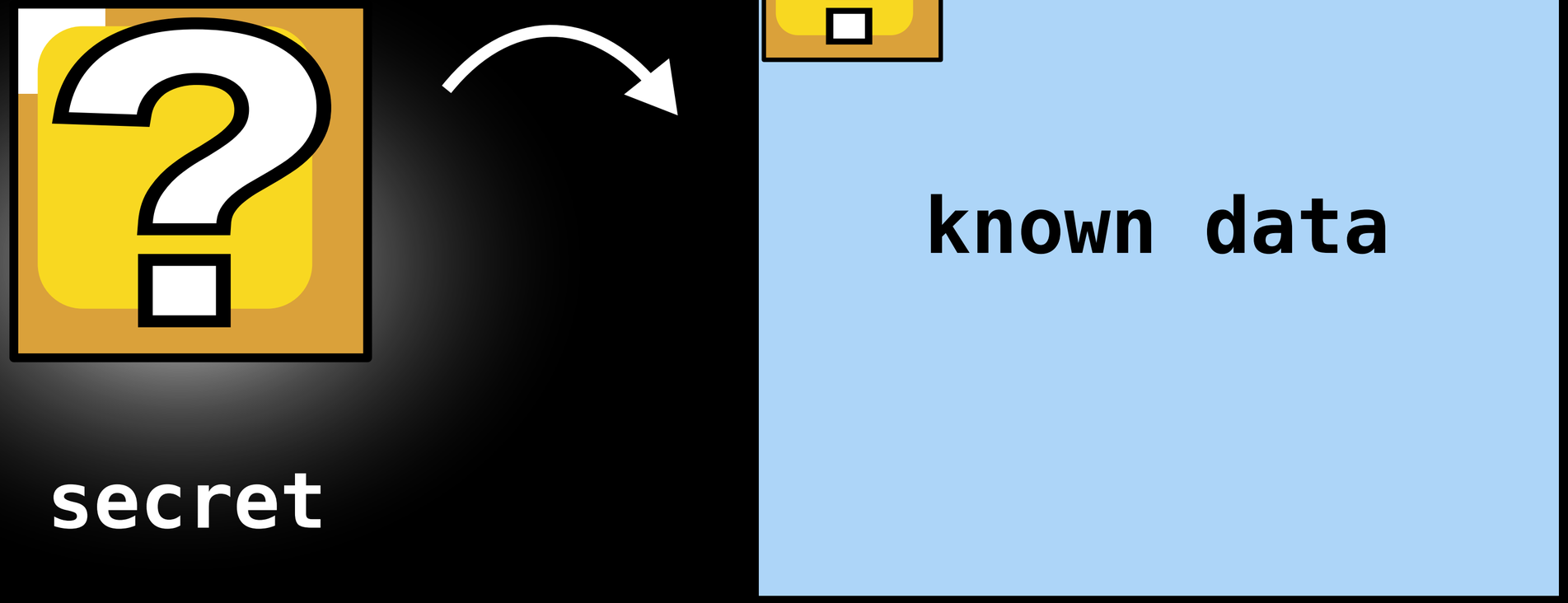
primitive #1: alignment probing



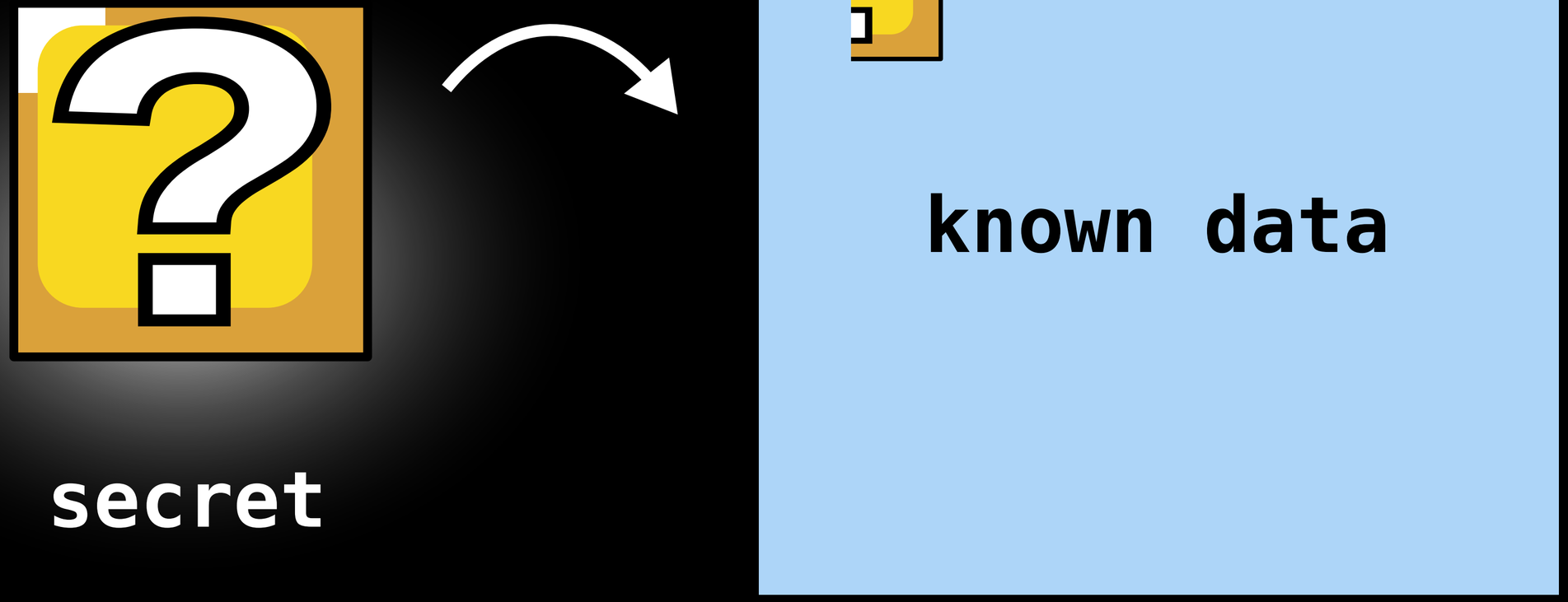
primitive #1: alignment probing



primitive #2: partial reuse



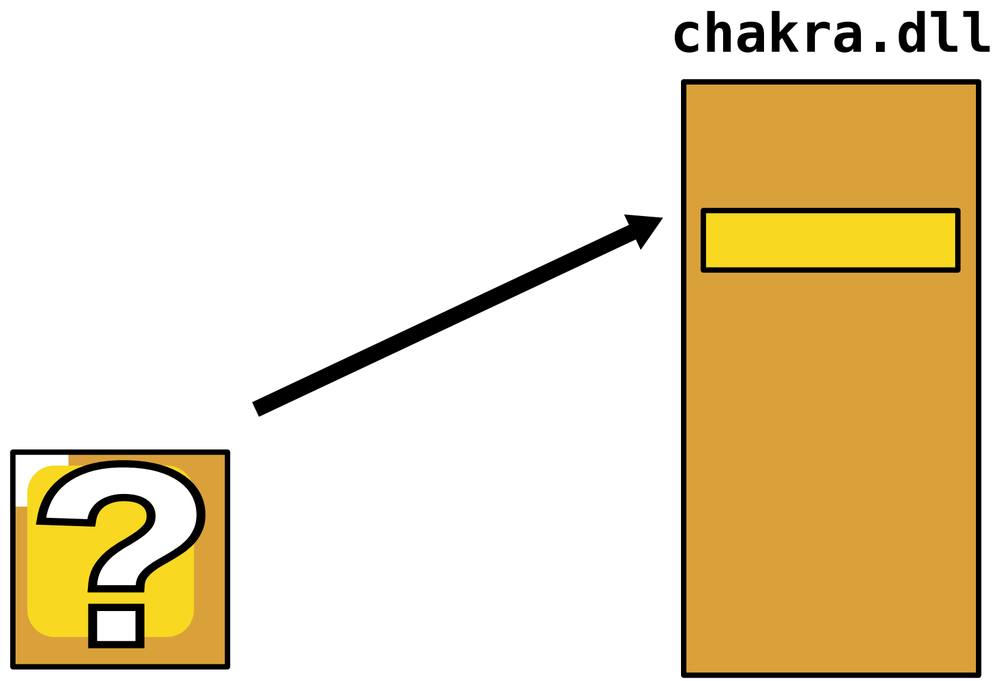
primitive #2: partial reuse



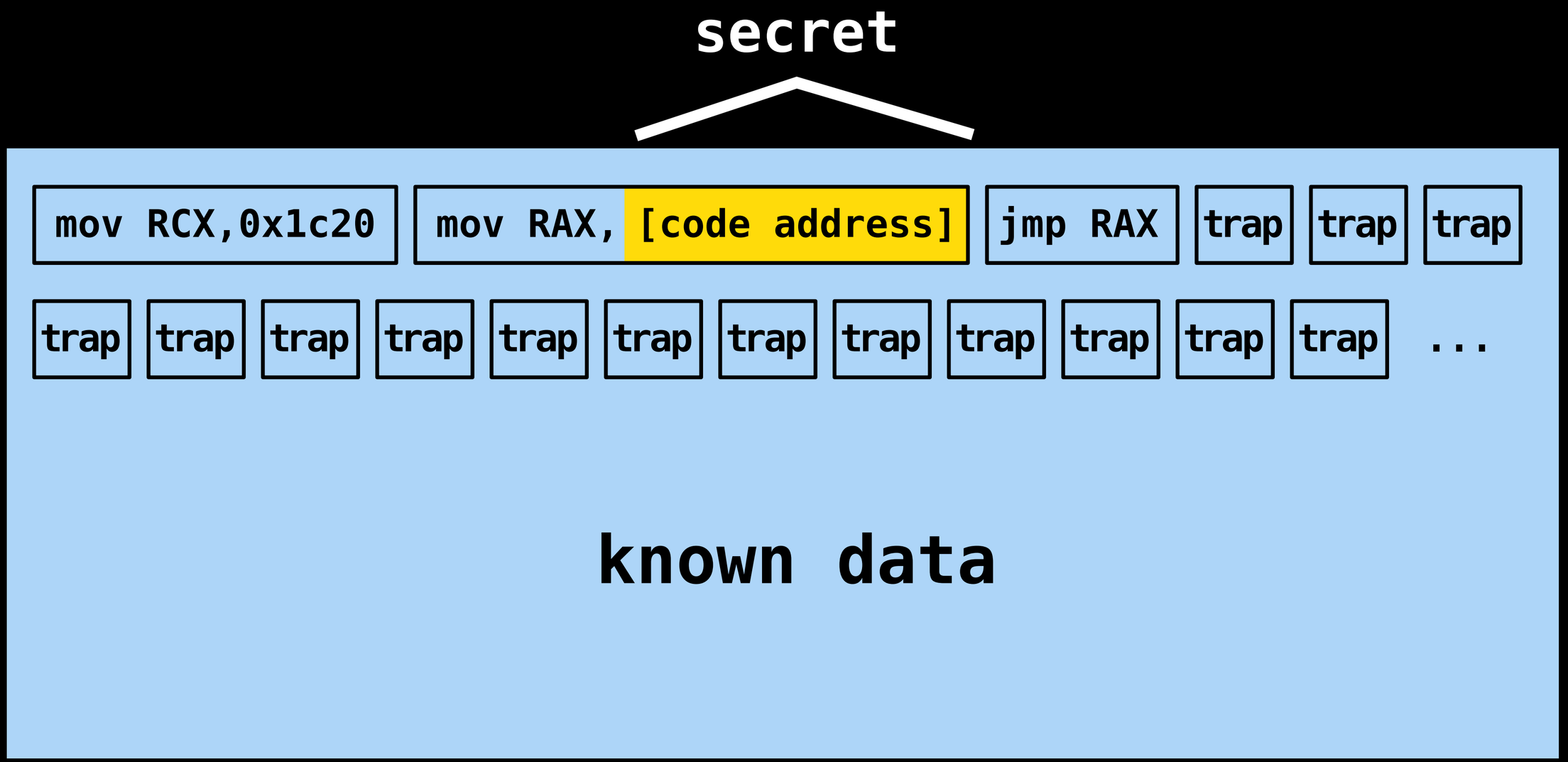
Outline:

Deduplication

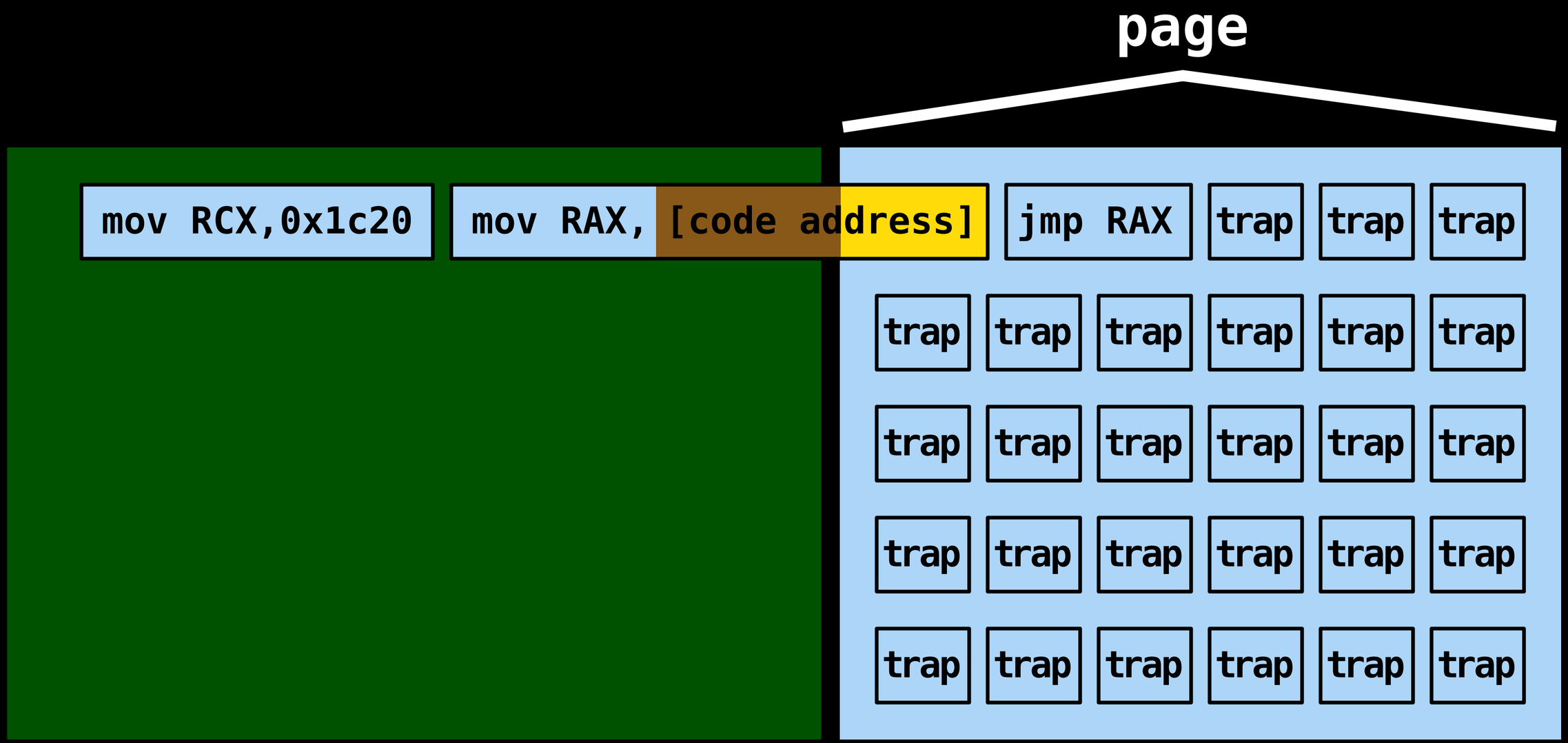
- leak heap & code addresses



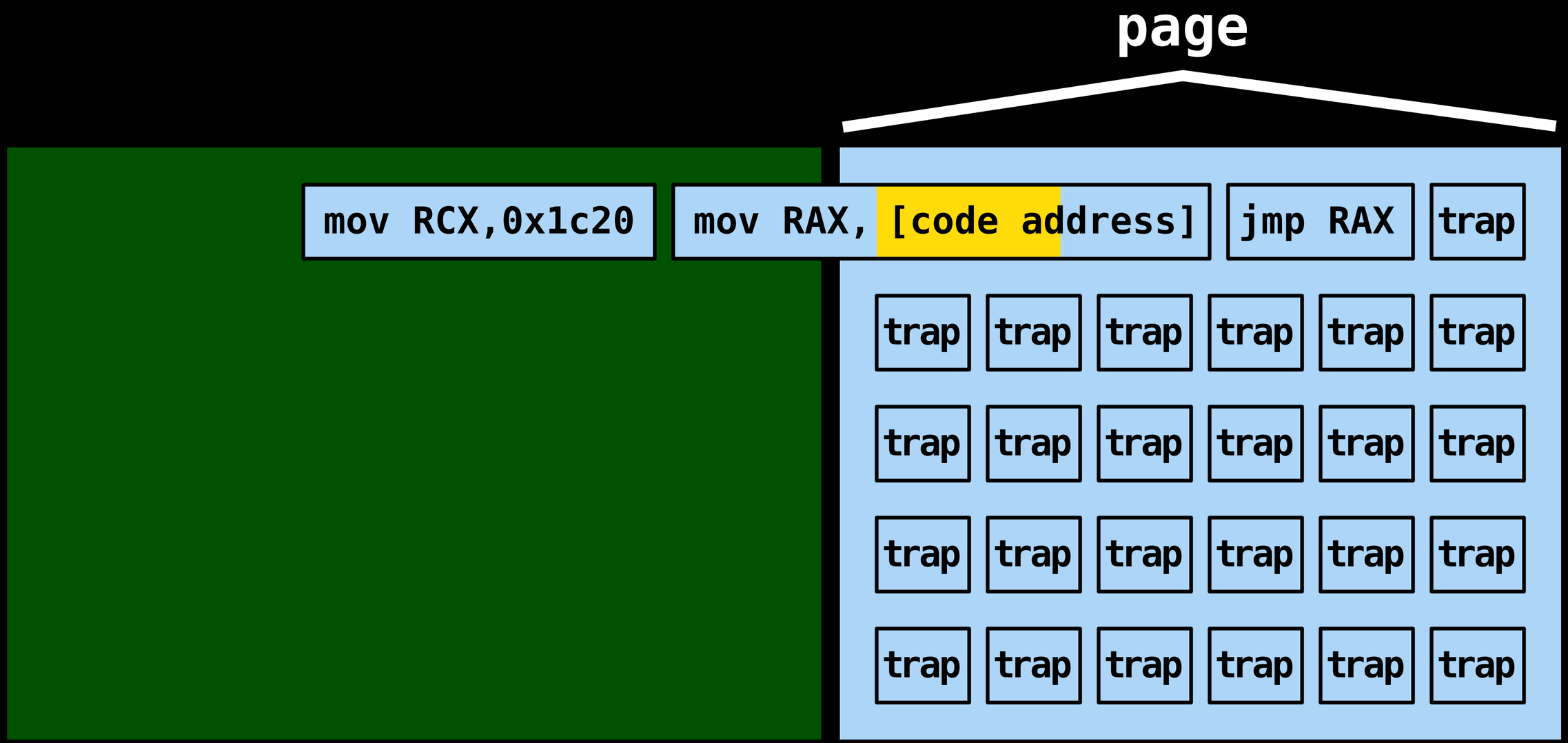
JIT function epilogue (MS Edge)



JIT function epilogue (MS Edge)



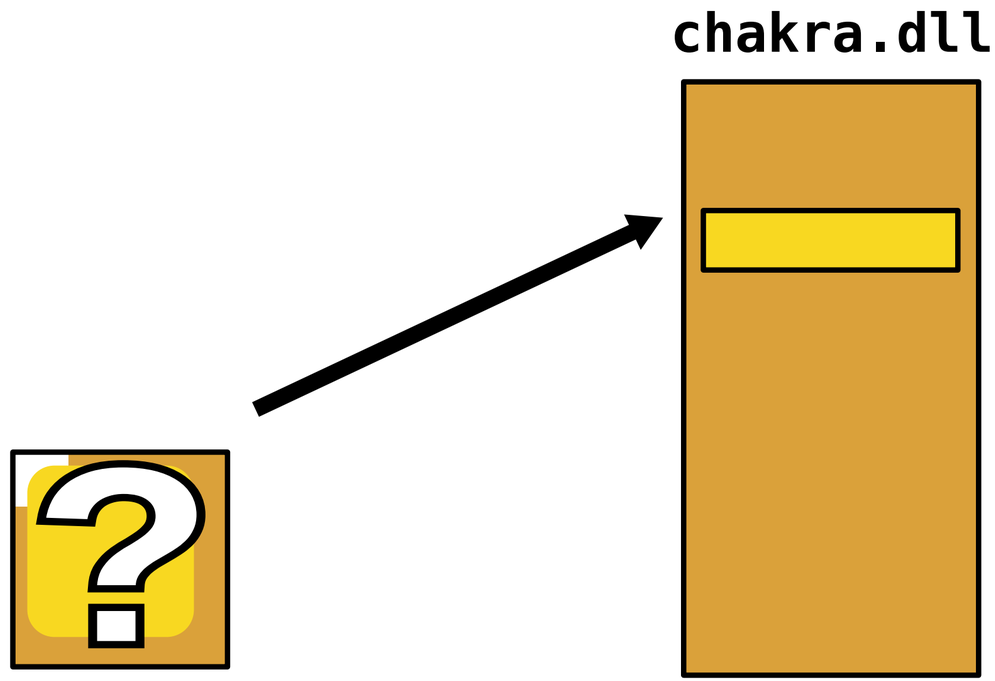
JIT function epilogue (MS Edge)



Outline:

Deduplication

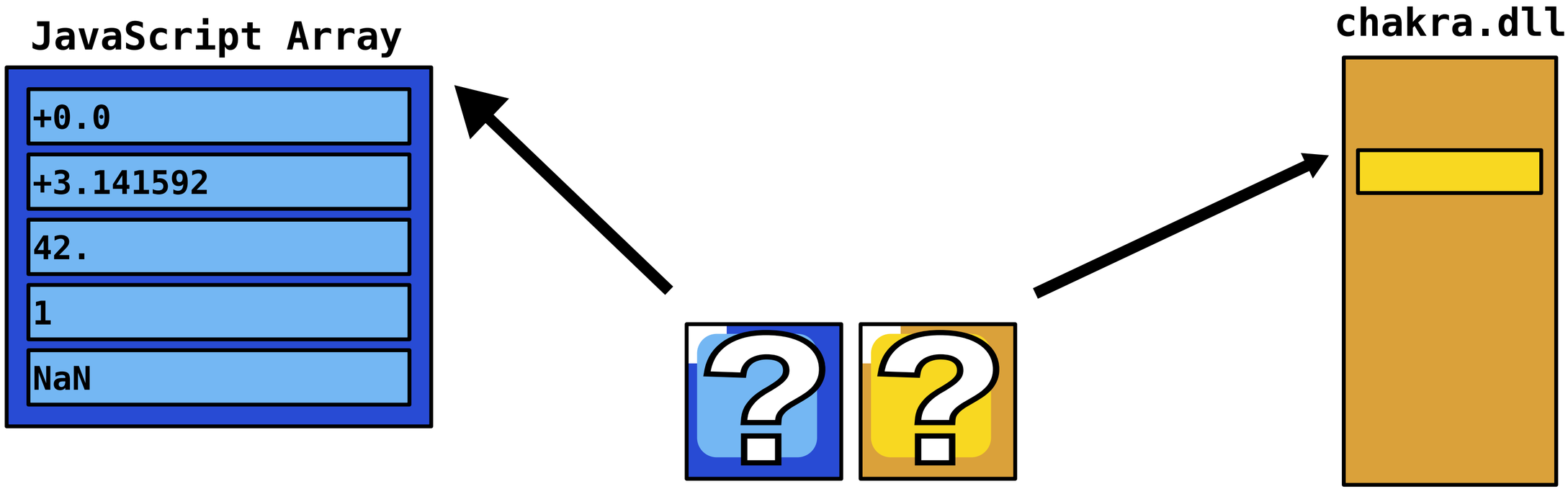
- leak heap & code addresses



Outline:

Deduplication

- leak heap & code addresses



We were not able to create pages leaking only part of our heap pointer.

Heap pointer entropy in Edge

0x5F48143540

Heap pointer entropy in Edge

advertised ASLR (24 bit)

0x5F48143540

Heap pointer entropy in Edge

advertised ASLR (24 bit)

0x5F48143540

non-deterministic bits
(+/- 36 bit)

Heap pointer entropy in Edge

64G

advertised ASLR (24 bit)

0x5F48143540

non-deterministic bits
(+/- 36 bit)

Heap pointer entropy in Edge

64G

advertised ASLR (24 bit)

0x5F48143540

256T

non-deterministic bits
(+/- 36 bit)

Heap pointer entropy in Edge

64G * redundancy

advertised ASLR (24 bit)

0x5F48143540

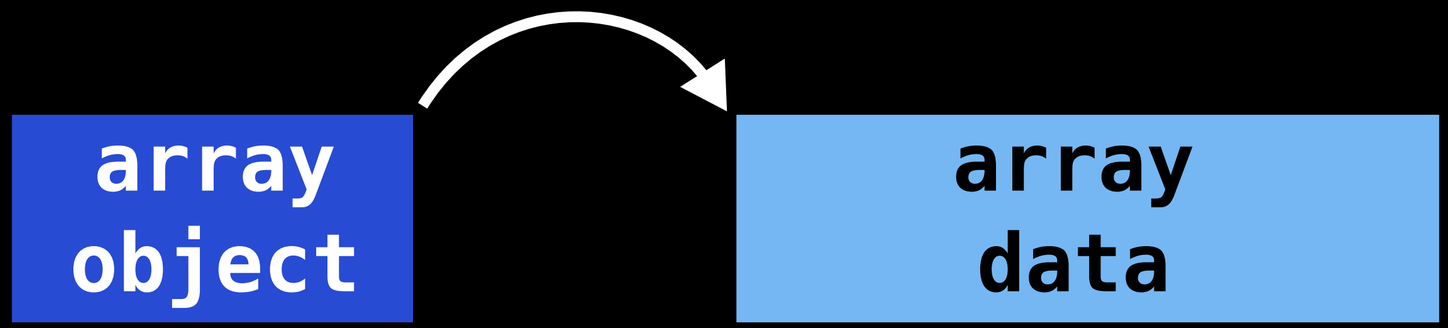
256T * redundancy

non-deterministic bits
(+/- 36 bit)

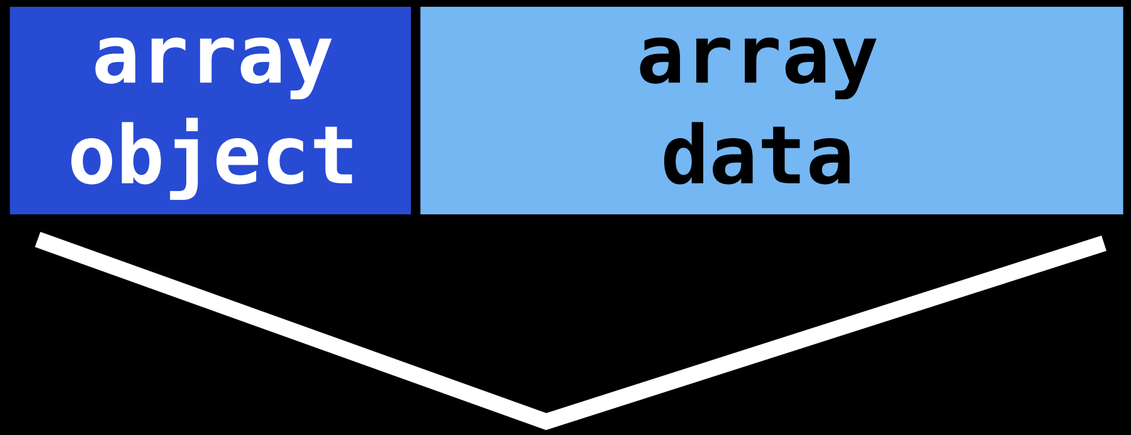
Slab allocator for JavaScript objects

array
object

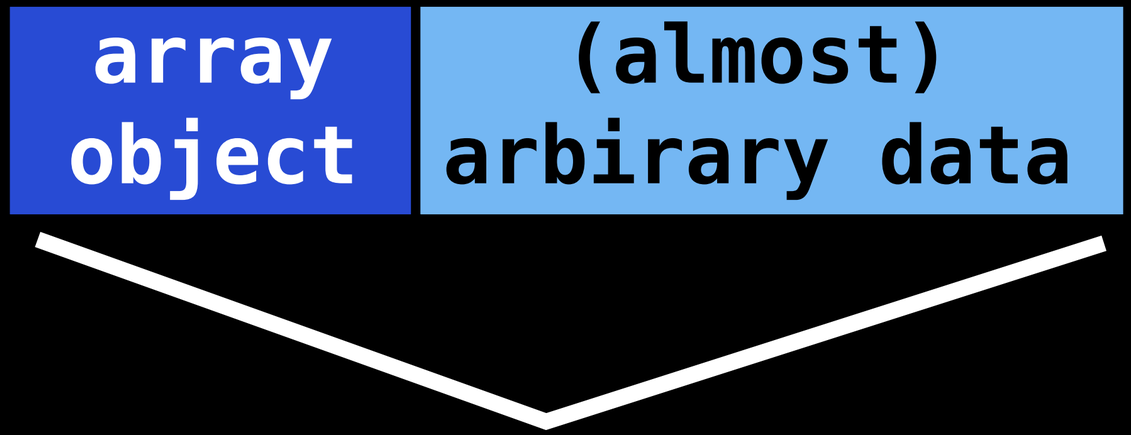
Slab allocator for JavaScript objects



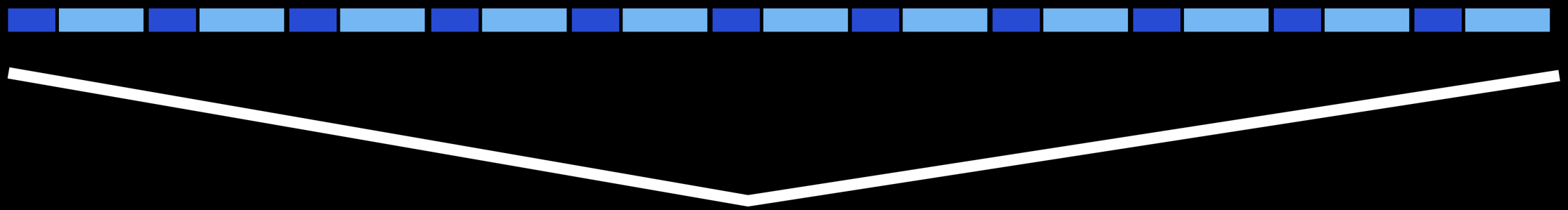
Slab allocator for JavaScript objects



Slab allocator for JavaScript objects

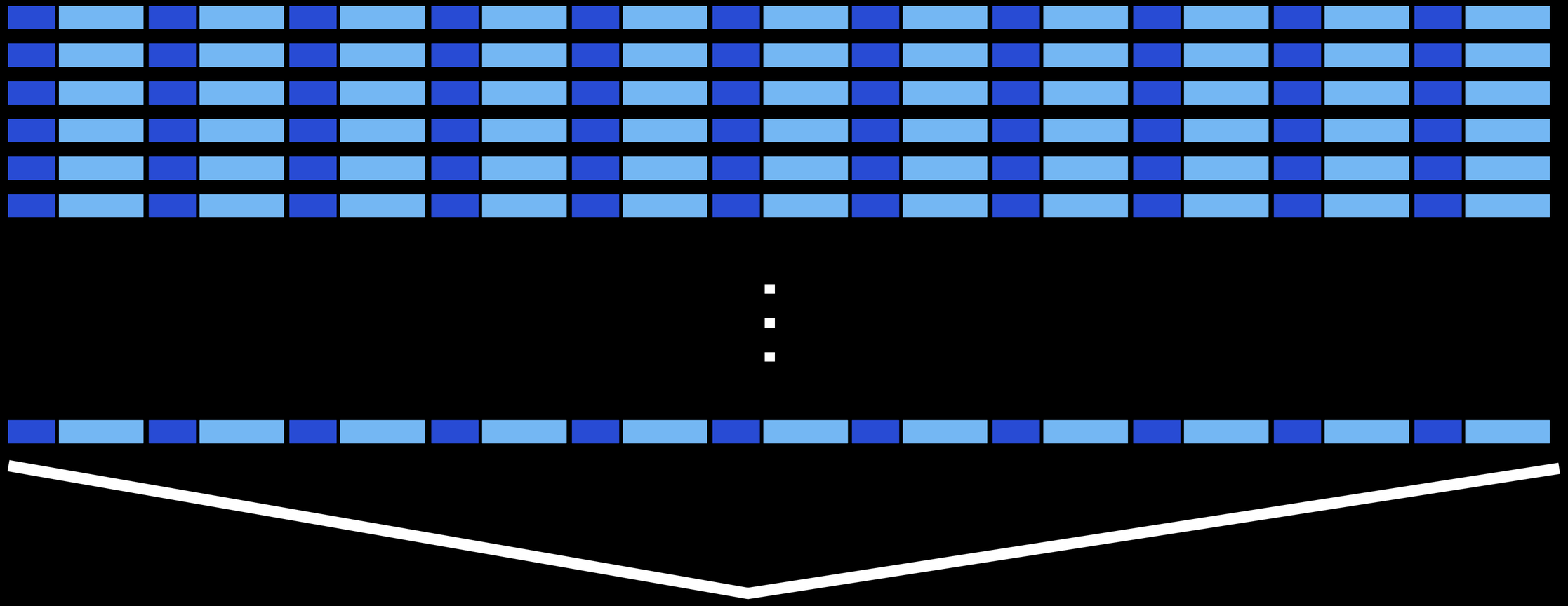


Slab allocator for JavaScript objects



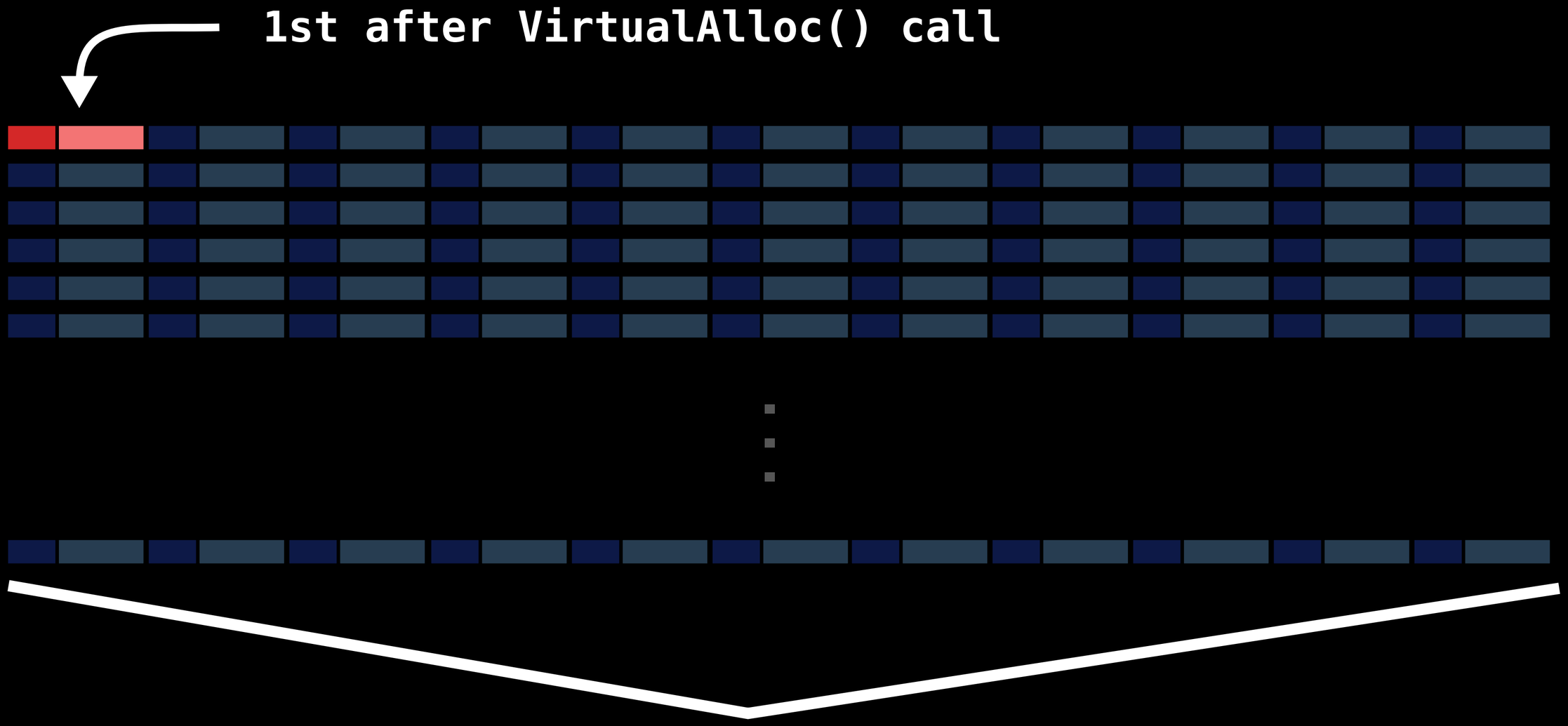
16K slab

Slab allocator for JavaScript objects

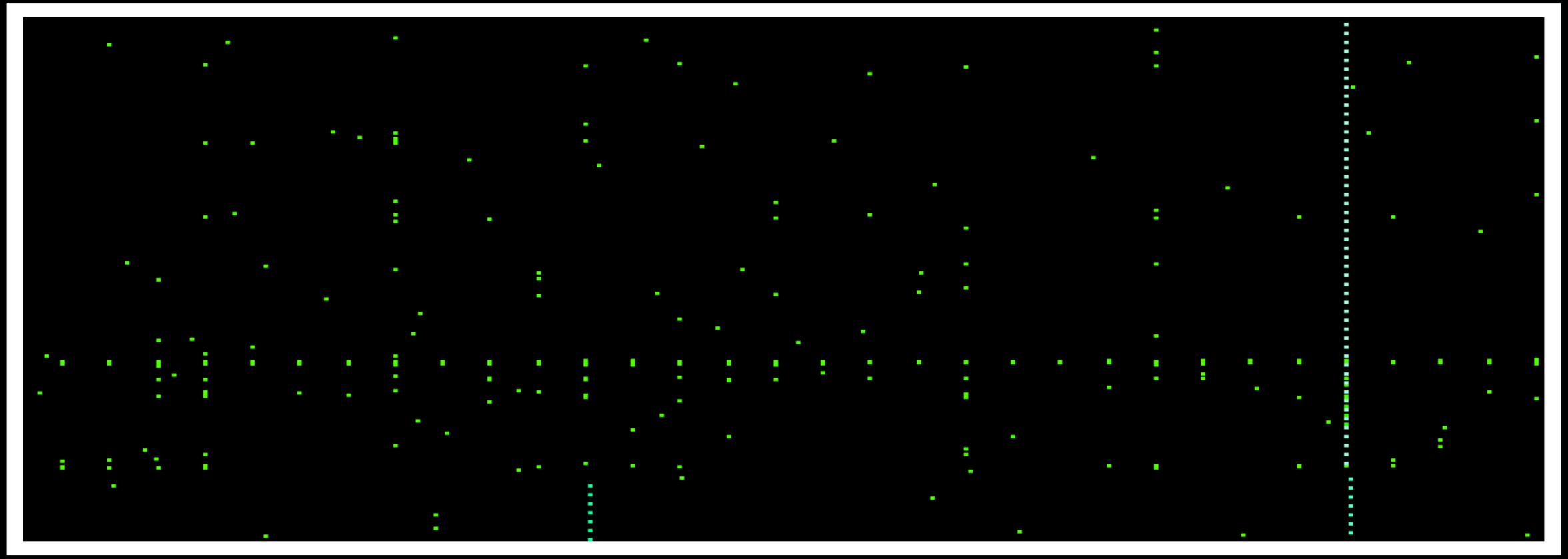


1M VirtualAlloc()

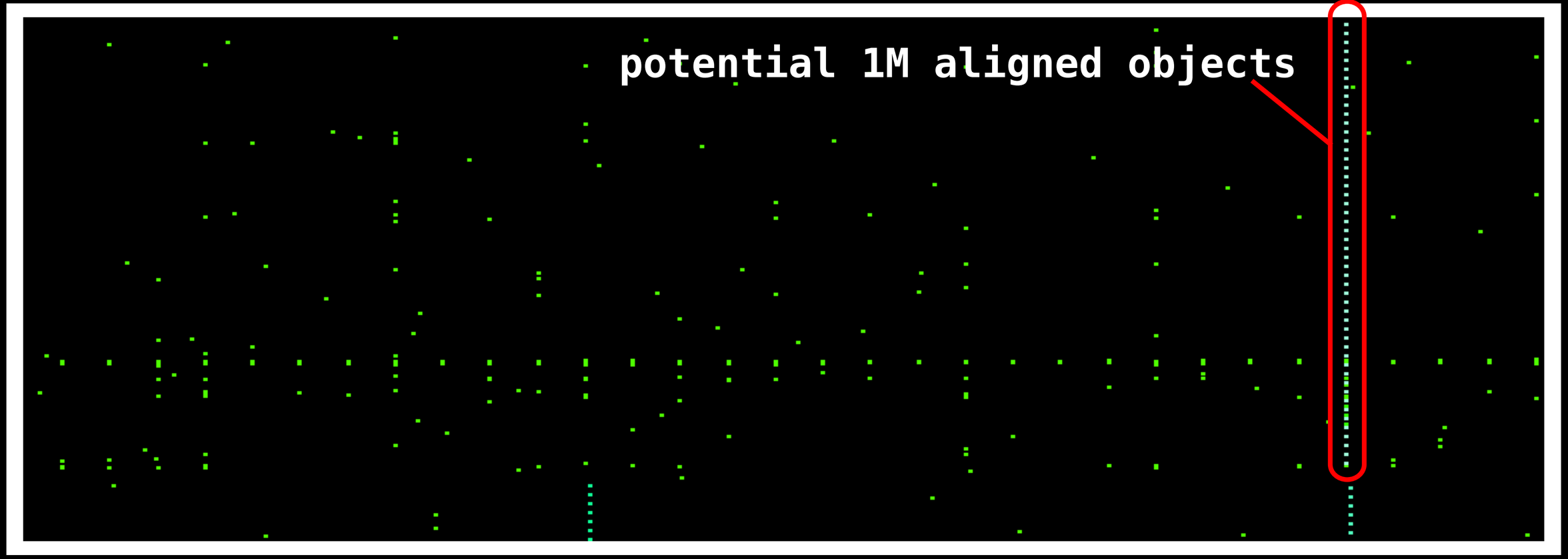
Slab allocator for JavaScript objects



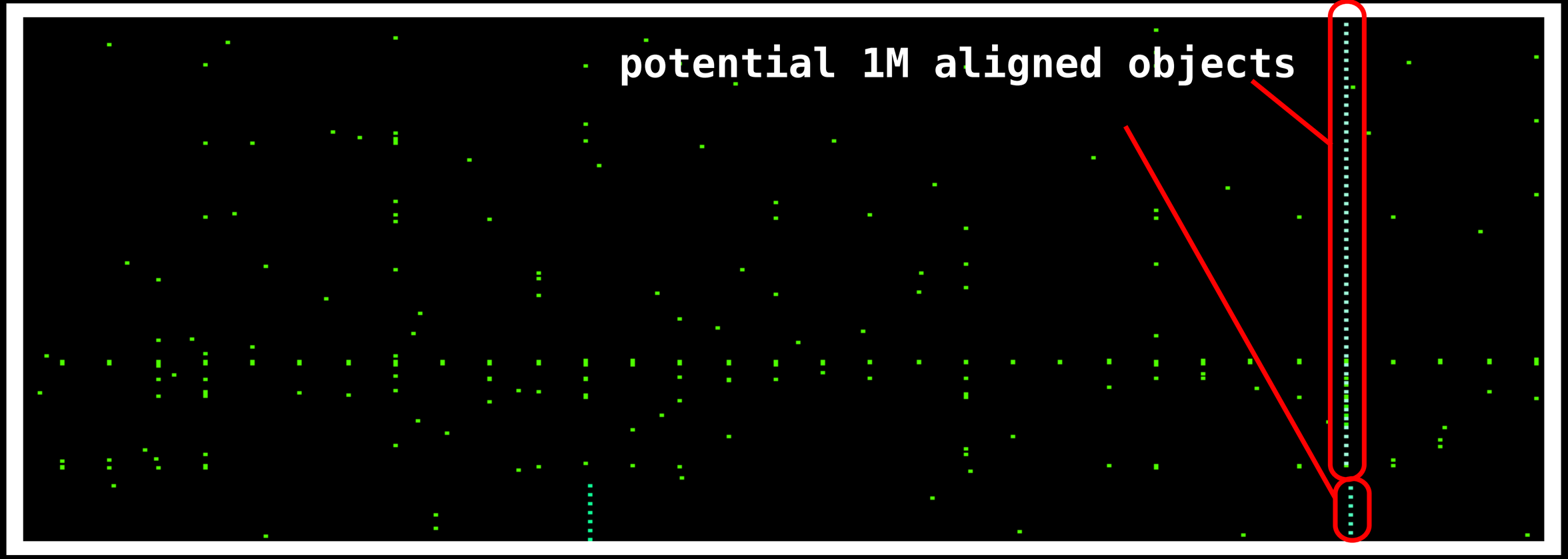
Slab allocator for JavaScript objects



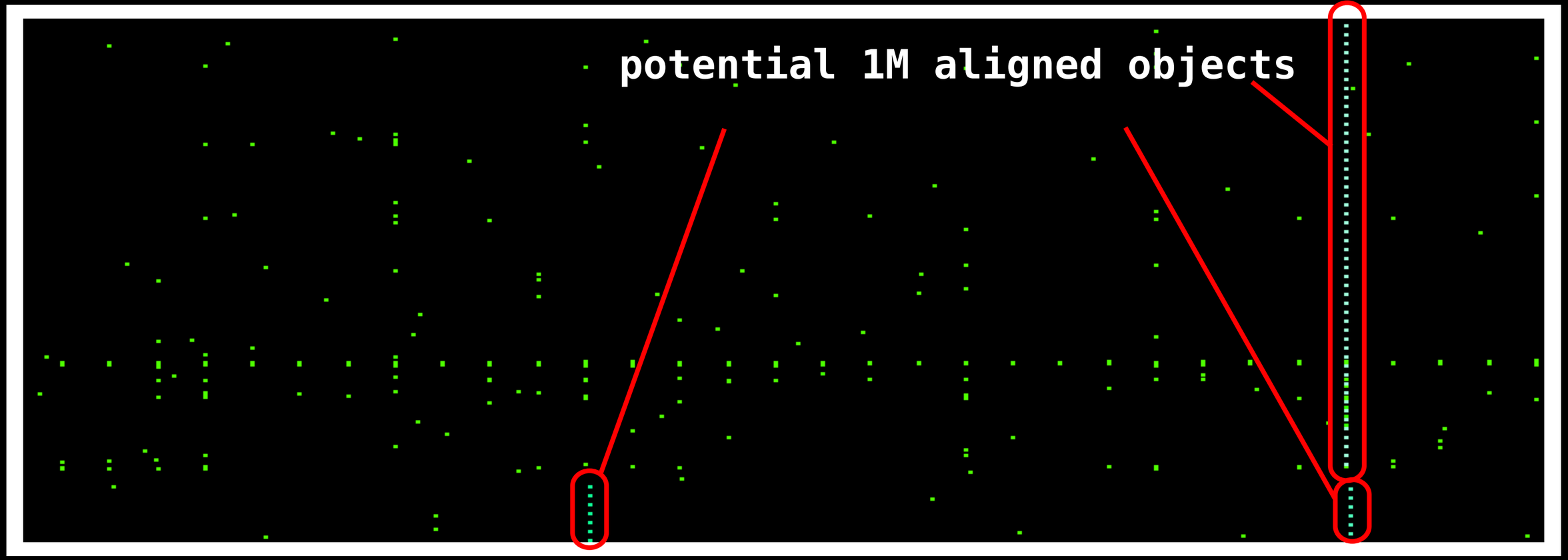
Slab allocator for JavaScript objects



Slab allocator for JavaScript objects



Slab allocator for JavaScript objects



Heap pointer entropy in Edge

64G

advertised ASLR (24 bit) * redundancy

0x5F48143540

256T

non-deterministic bits * redundancy
(+/- 36 bit)

Heap pointer entropy in Edge

64G

advertised ASLR (24 bit) * redundancy

0x5F48100000

entropy after 1MB alignment
(20 bit)

Heap pointer entropy in Edge

64G

advertised ASLR (24 bit) * redundancy

0x5F48100000

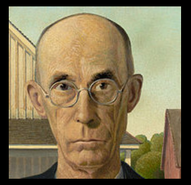
4G

entropy after 1MB alignment * redundancy
(20 bit)

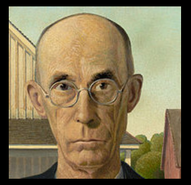
birthday problem

birthday problem

birthday problem



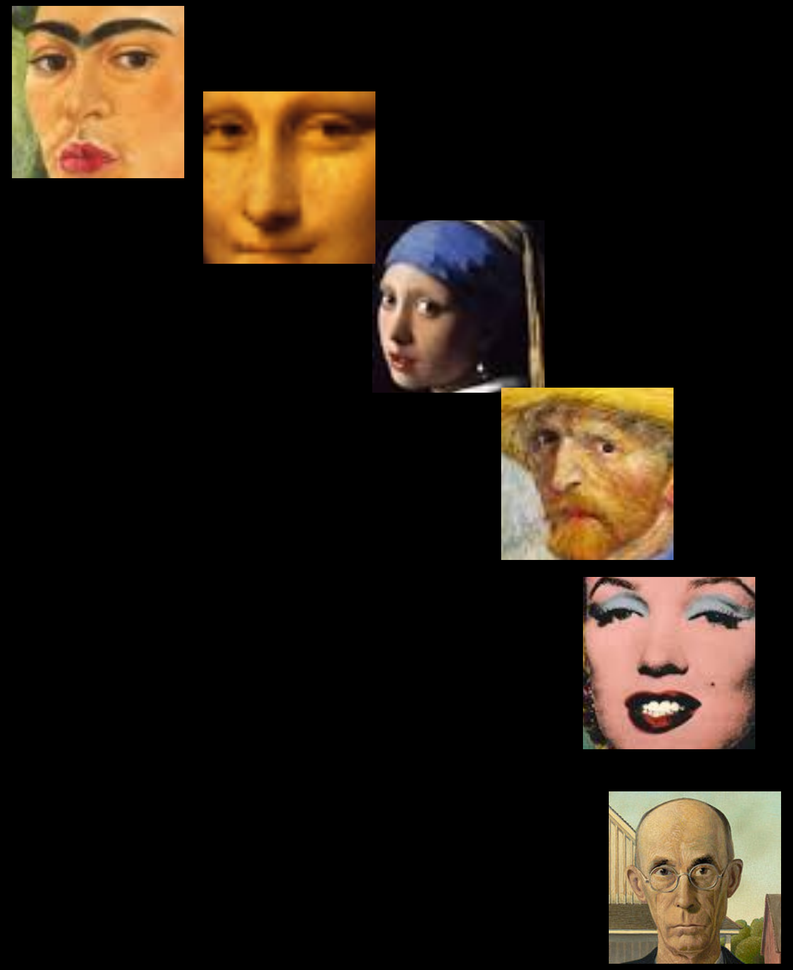
birthday problem



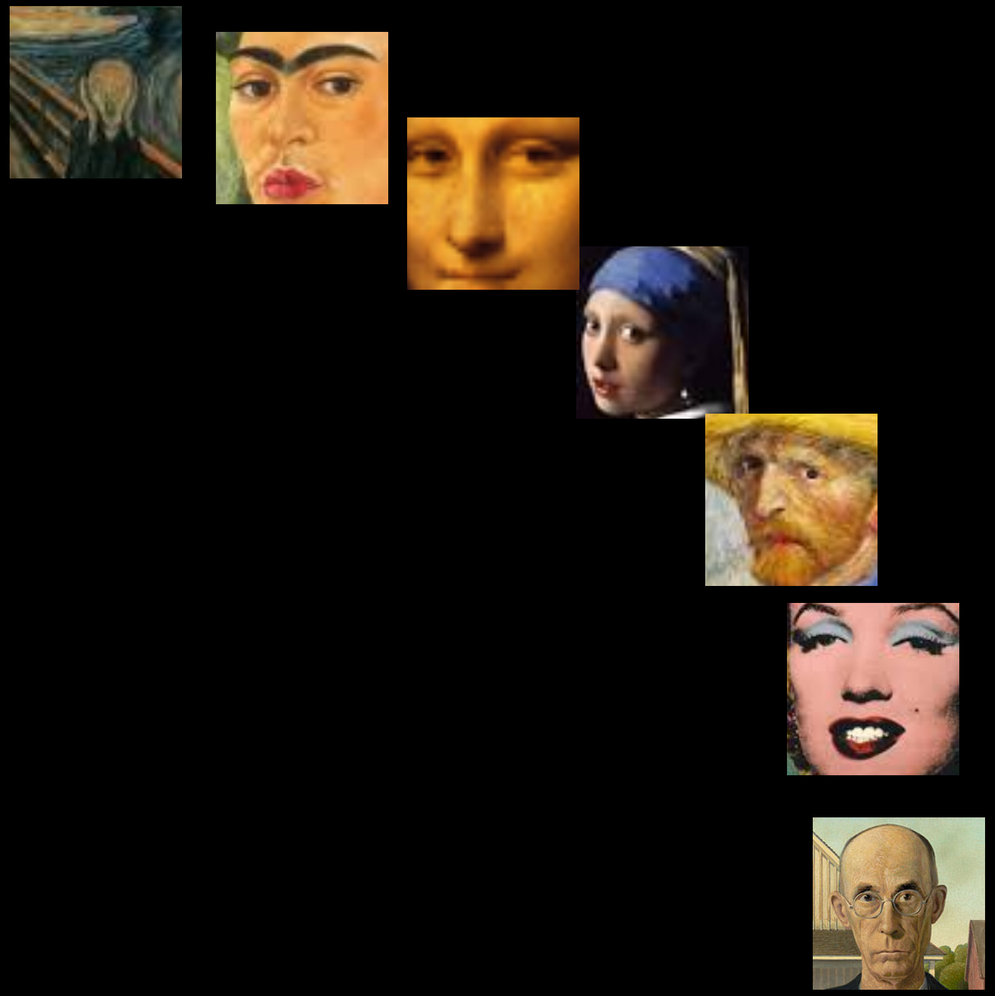
birthday problem

birthday problem

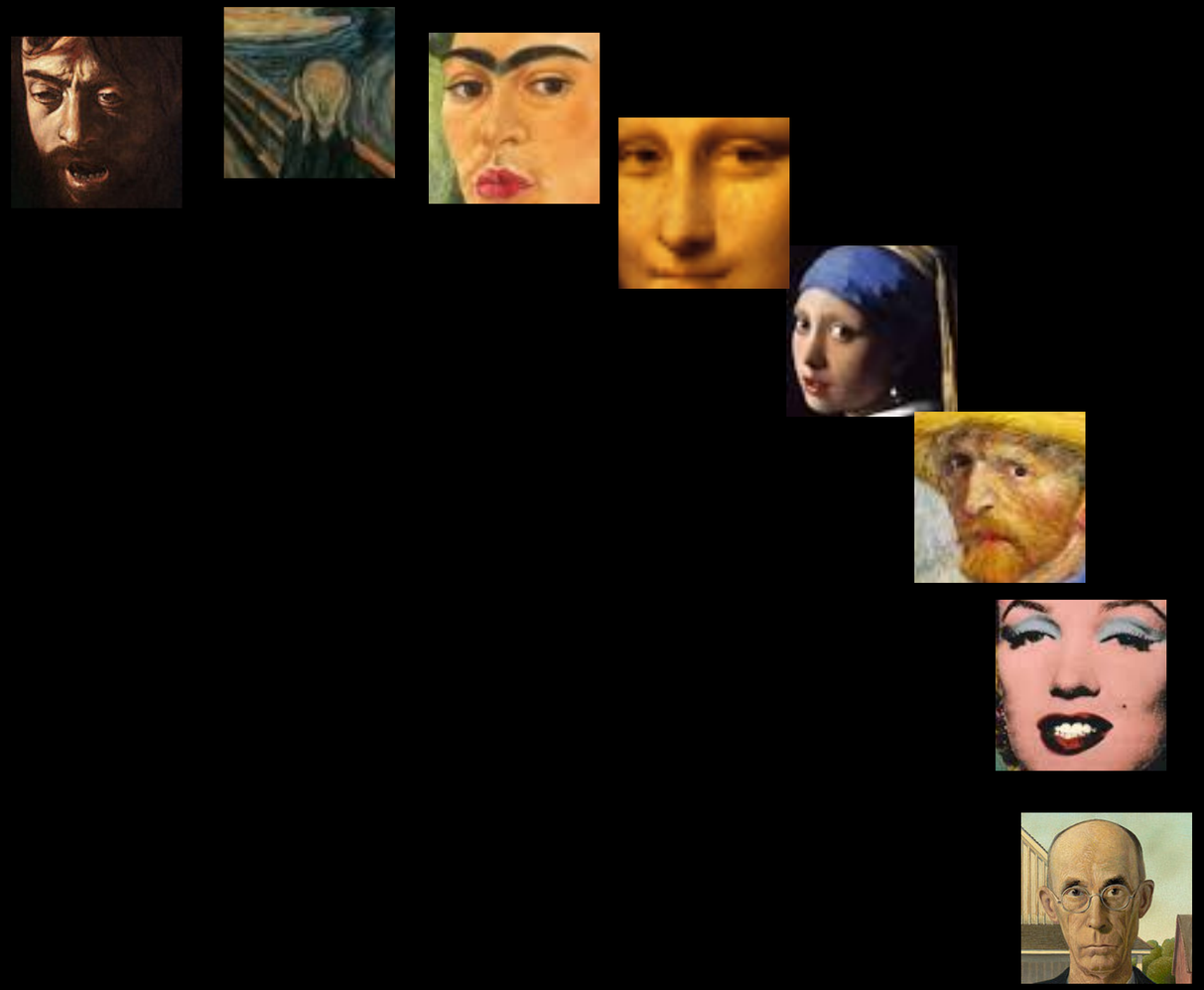
birthday problem



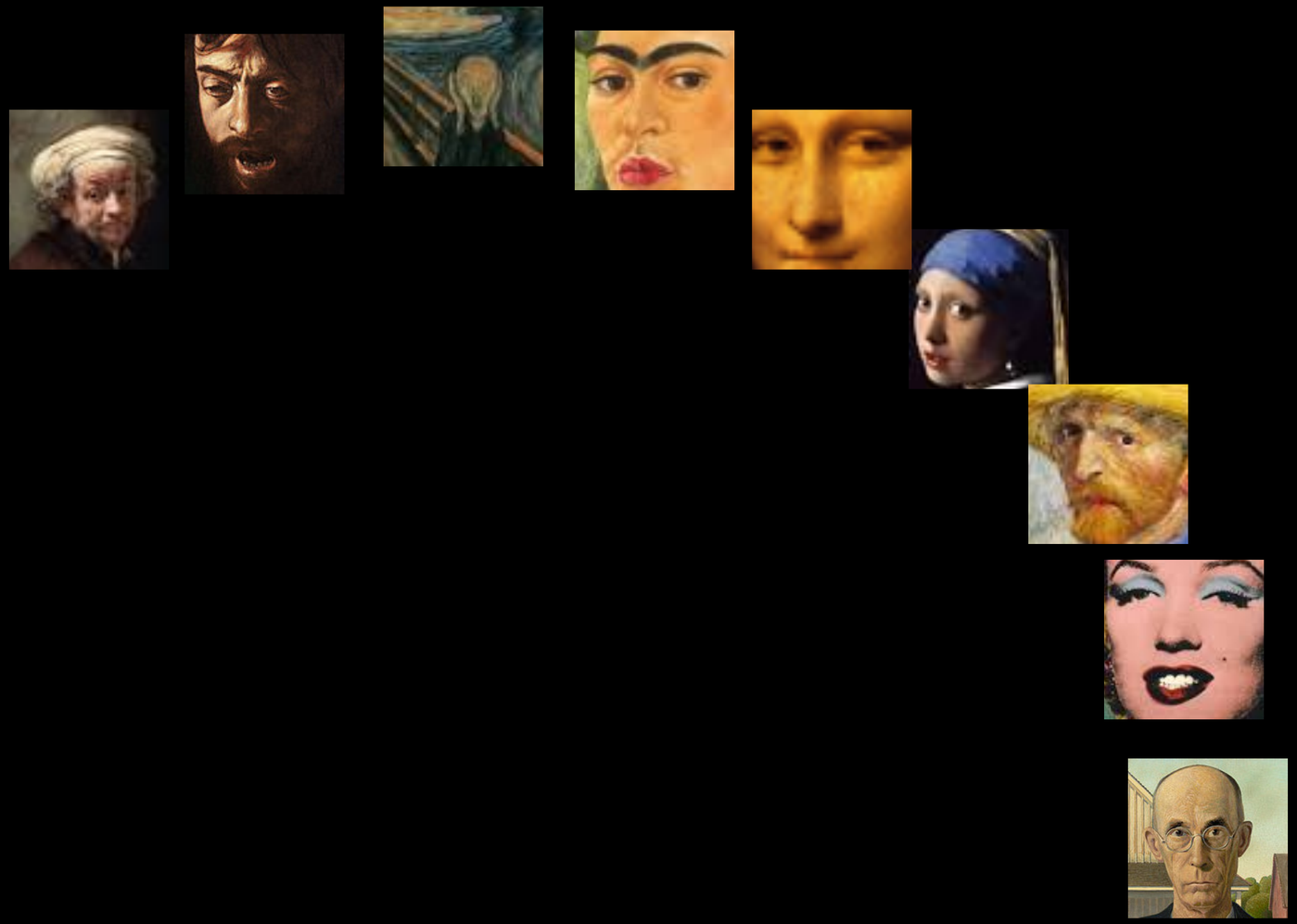
birthday problem



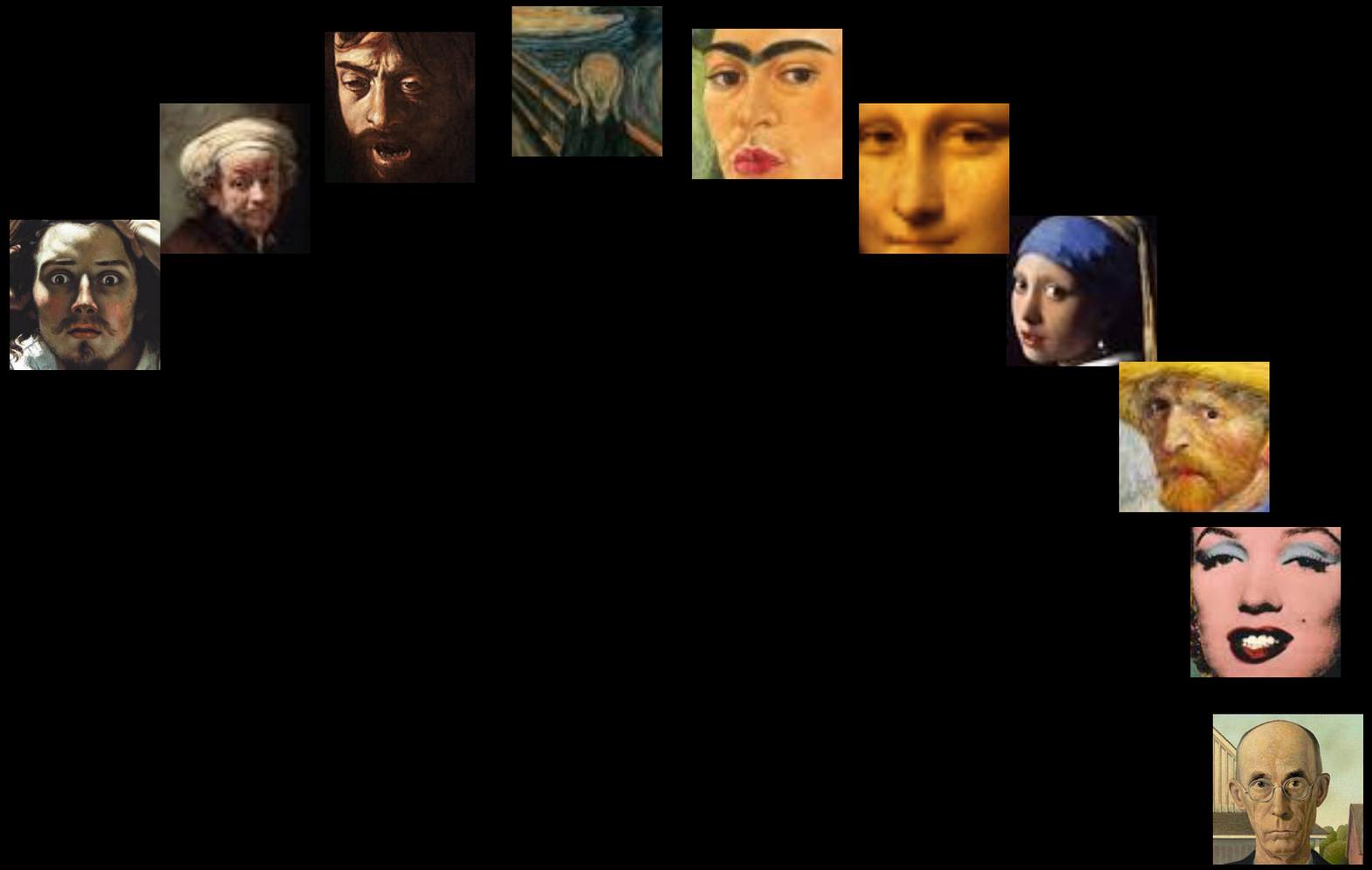
birthday problem



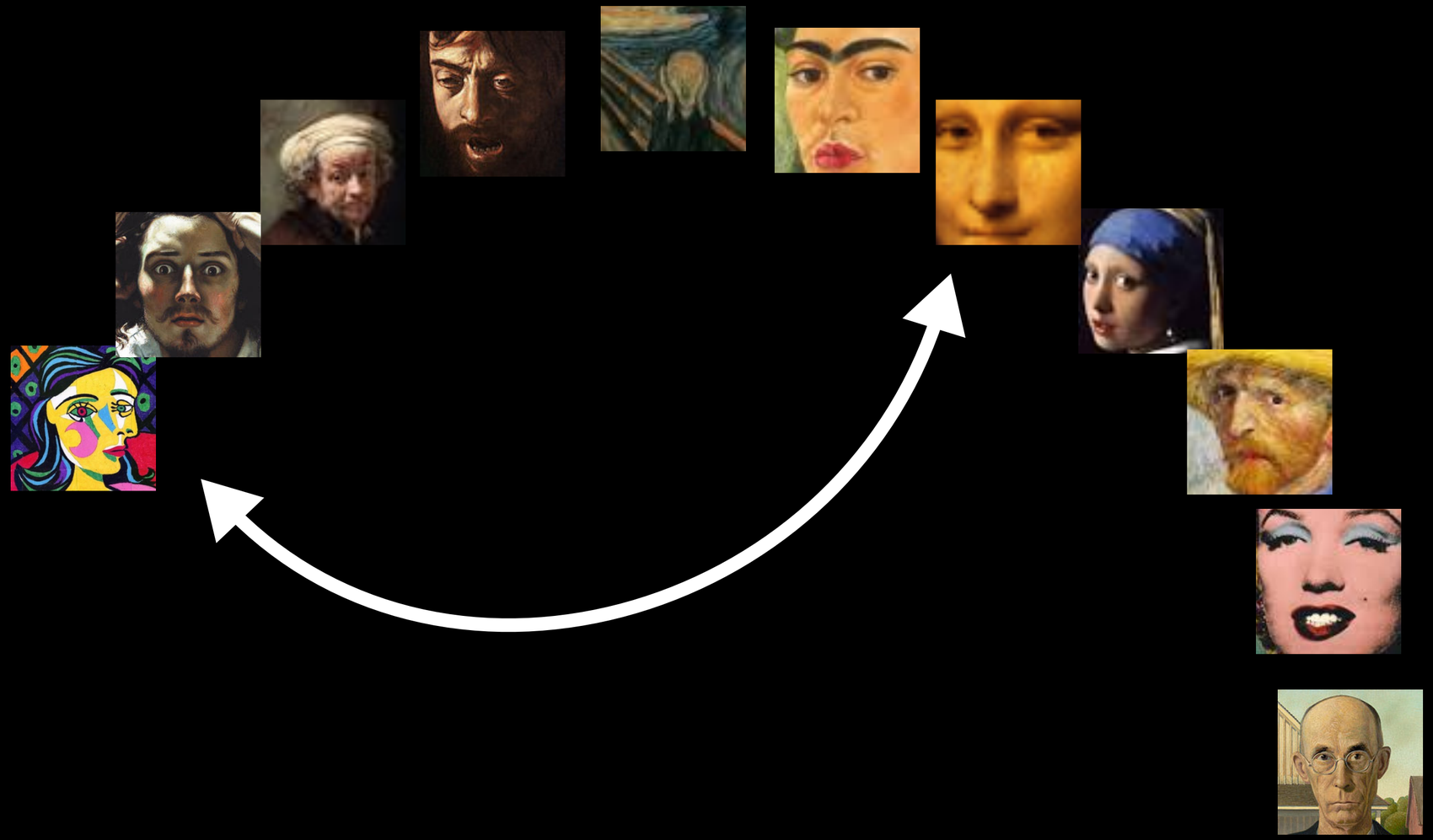
birthday problem



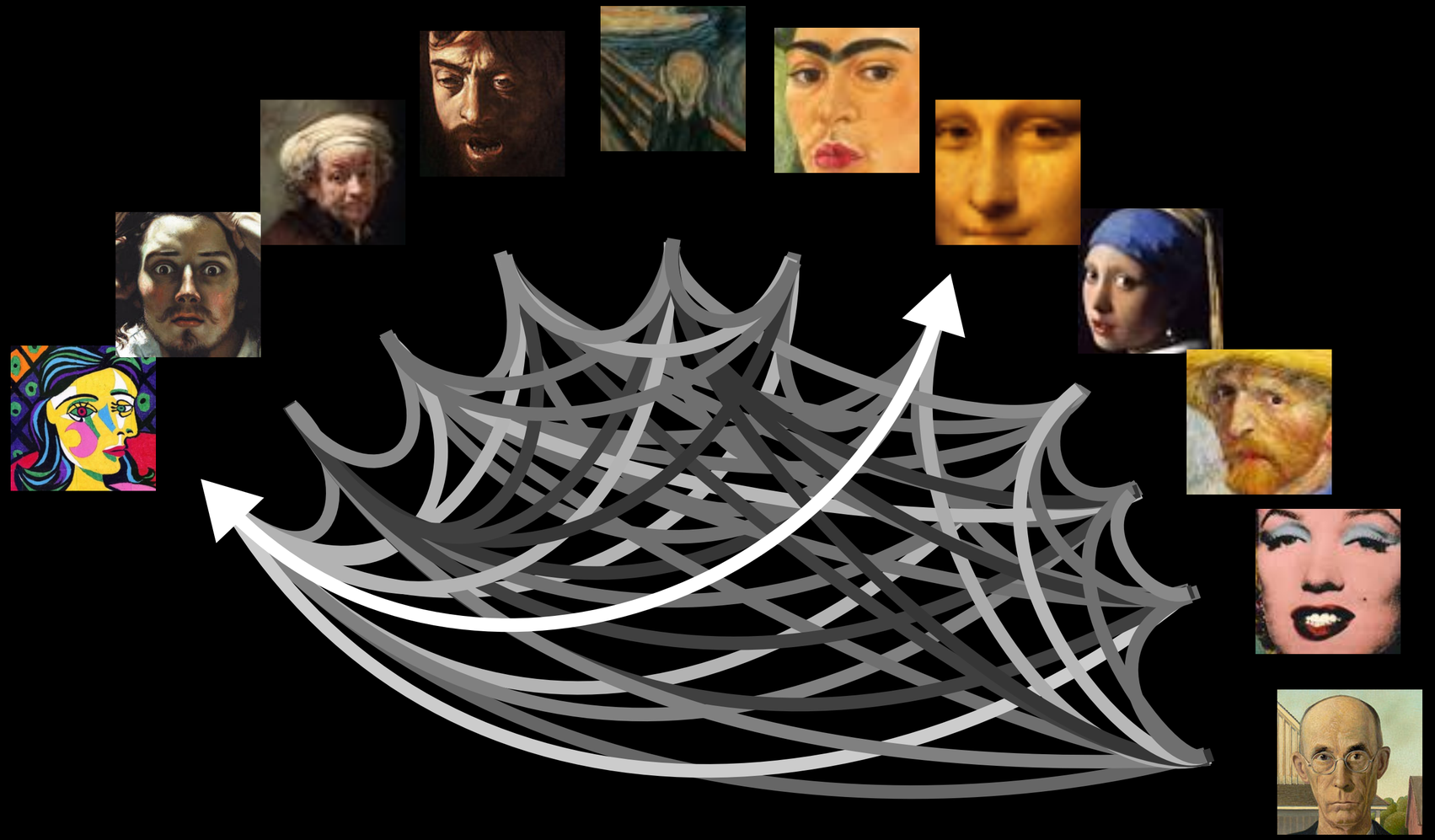
birthday problem



birthday problem

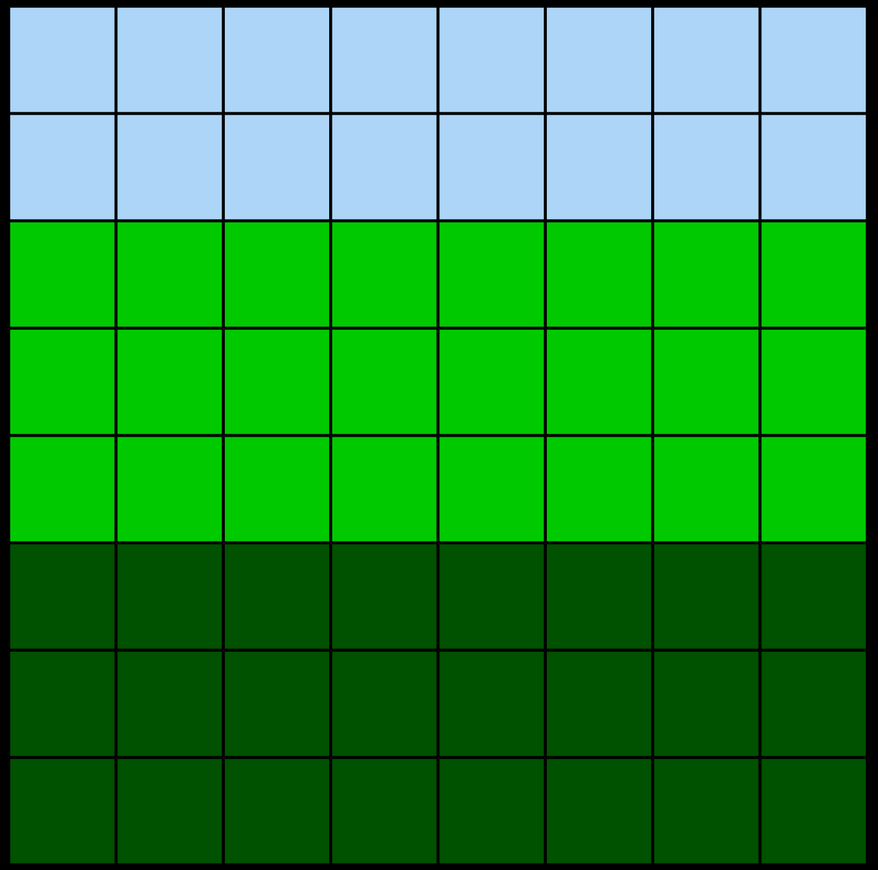


birthday problem

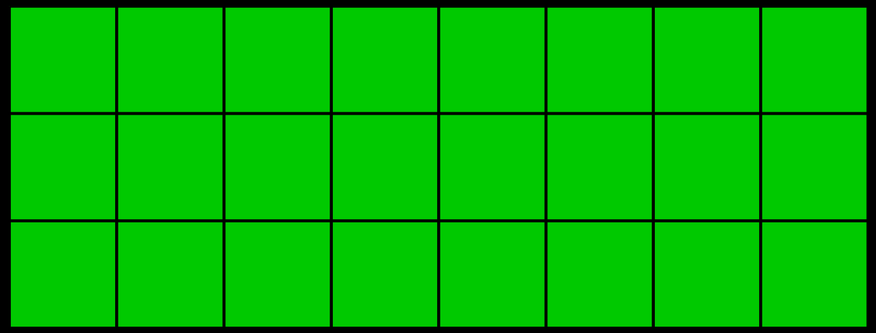


primitive #3: birthday heapspray

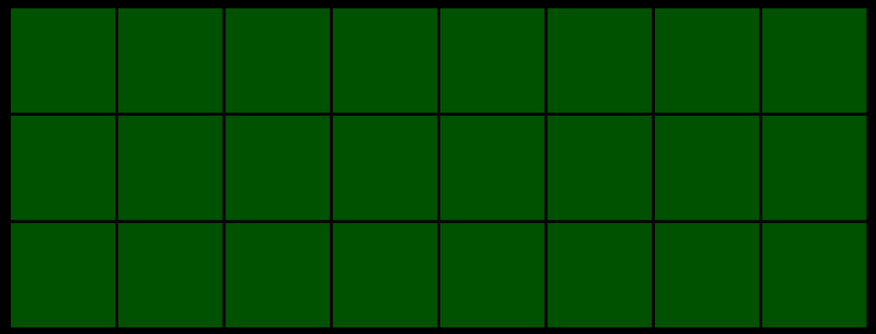
physical memory



attacker memory

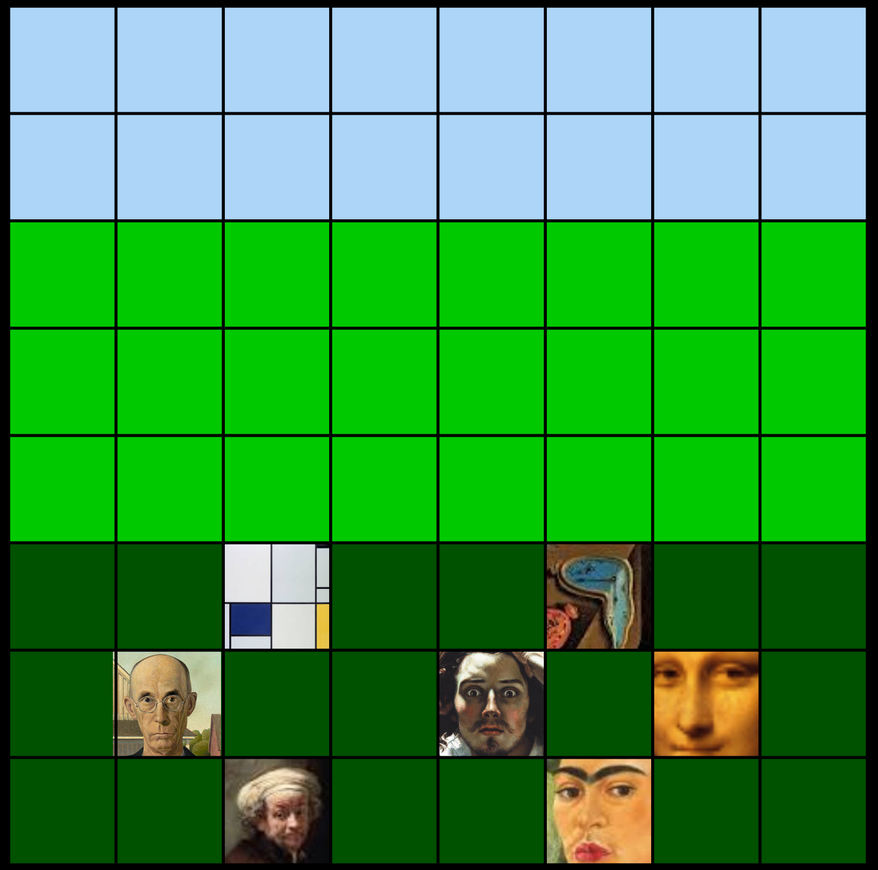


victim memory

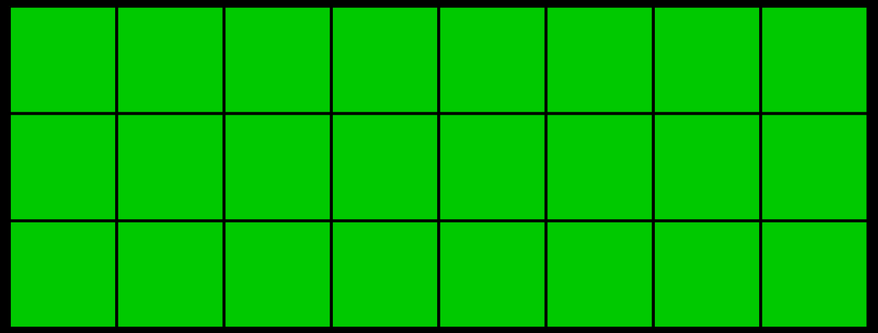


primitive #3: birthday heapspray

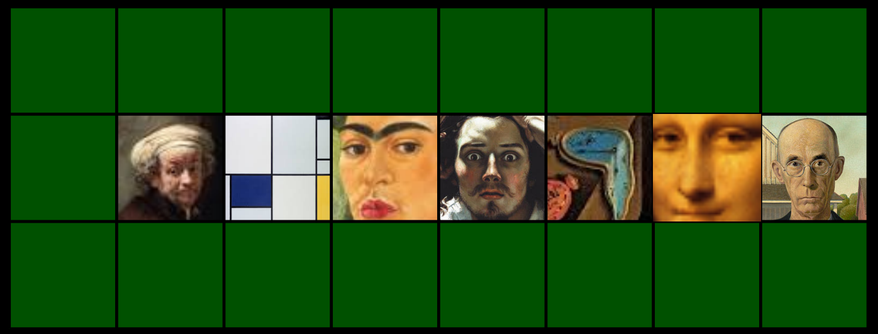
physical memory



attacker memory

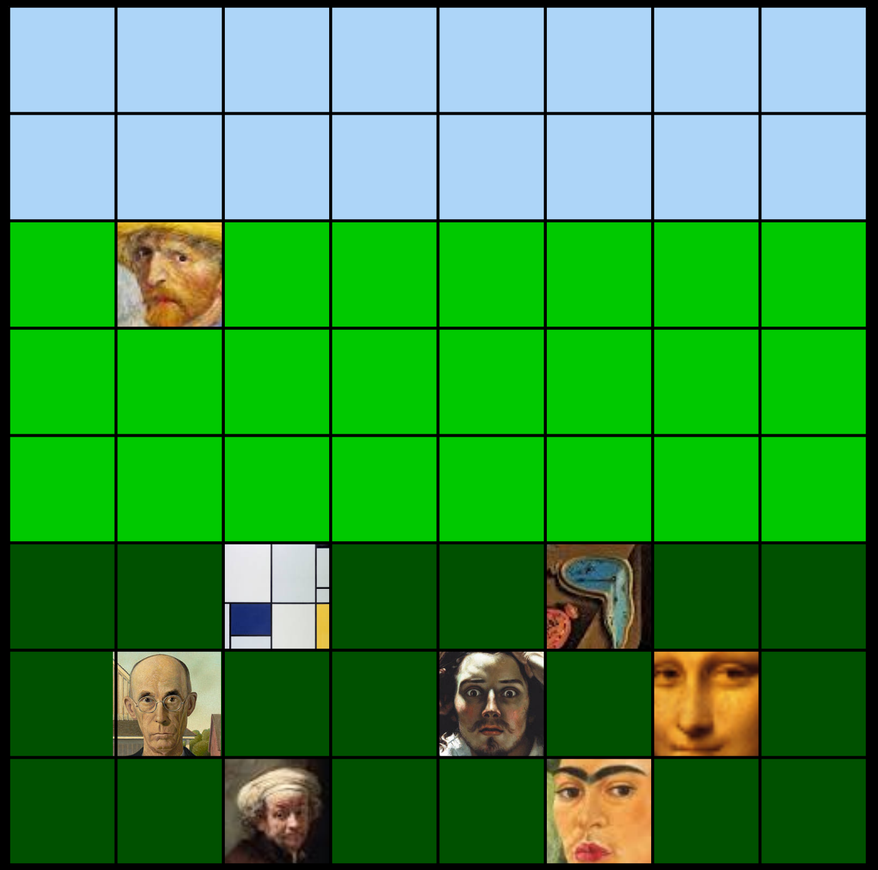


victim memory

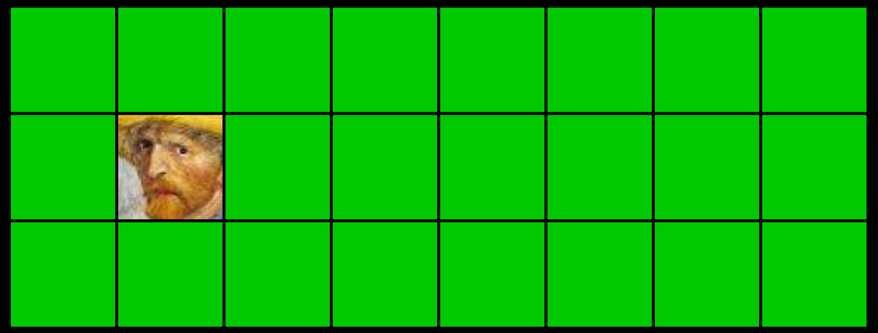


primitive #3: birthday heapspray

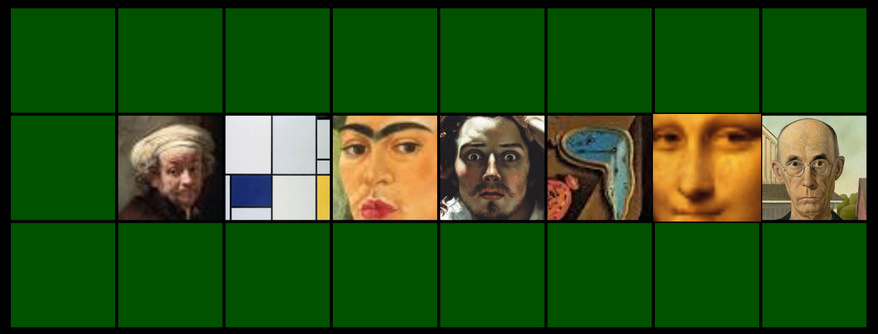
physical memory



attacker memory

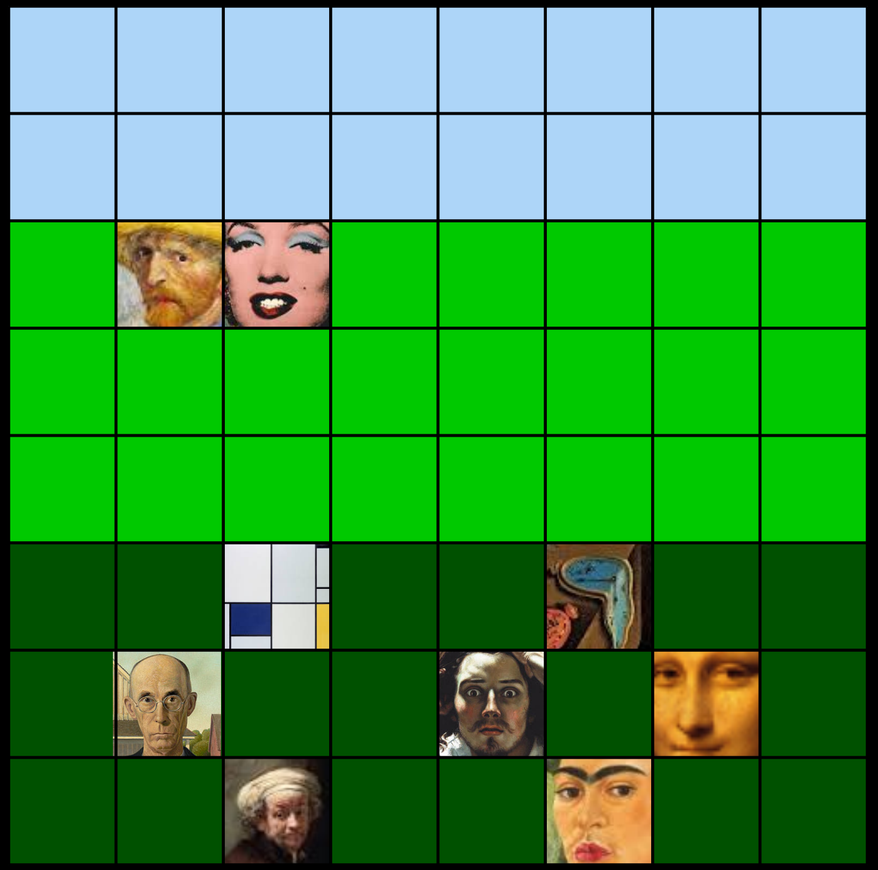


victim memory

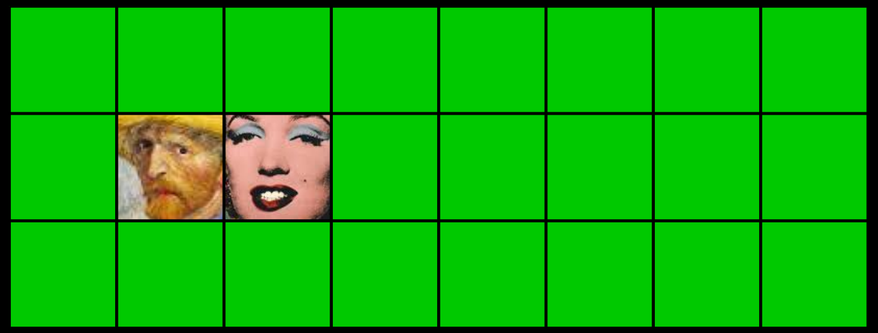


primitive #3: birthday heapspray

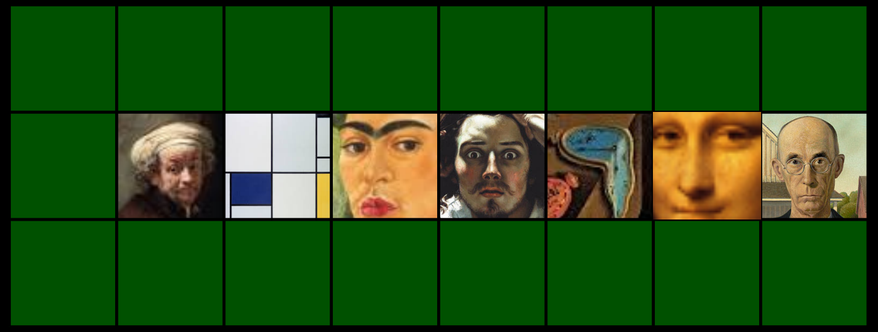
physical memory



attacker memory

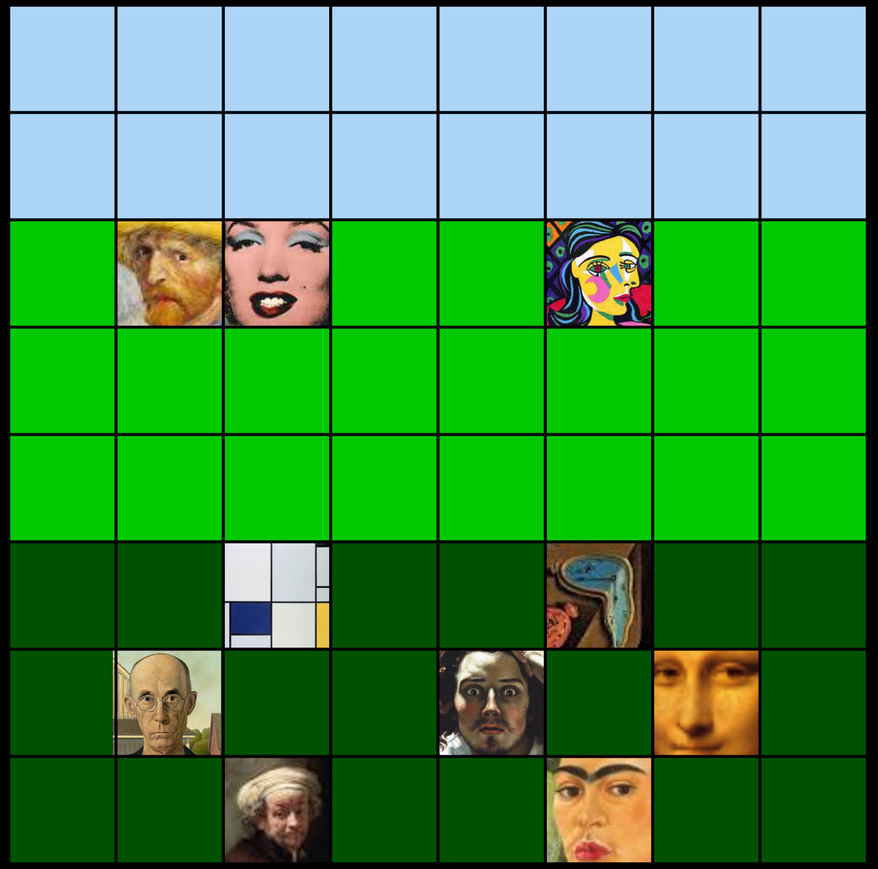


victim memory

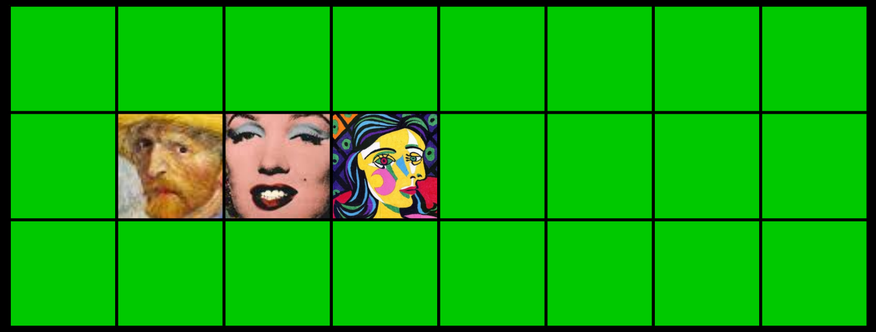


primitive #3: birthday heapspray

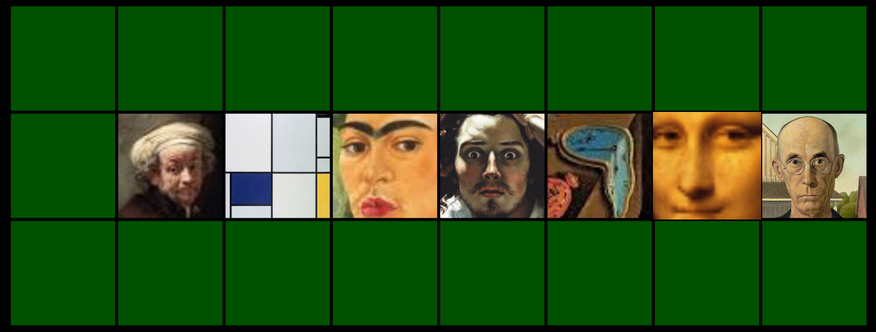
physical memory



attacker memory

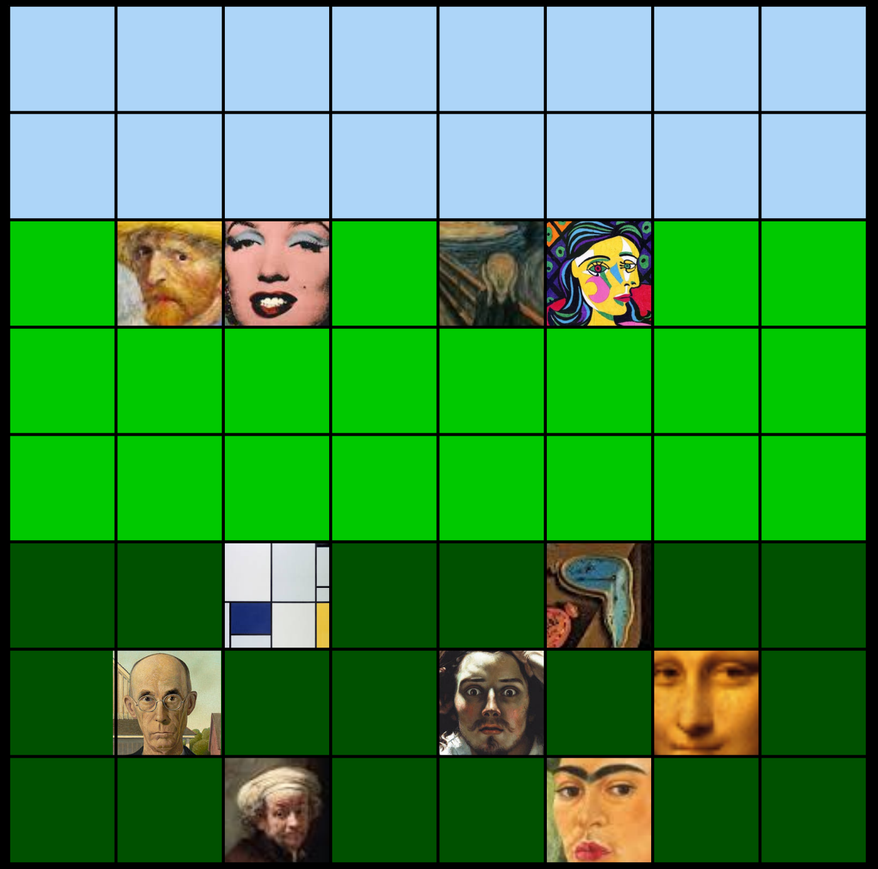


victim memory

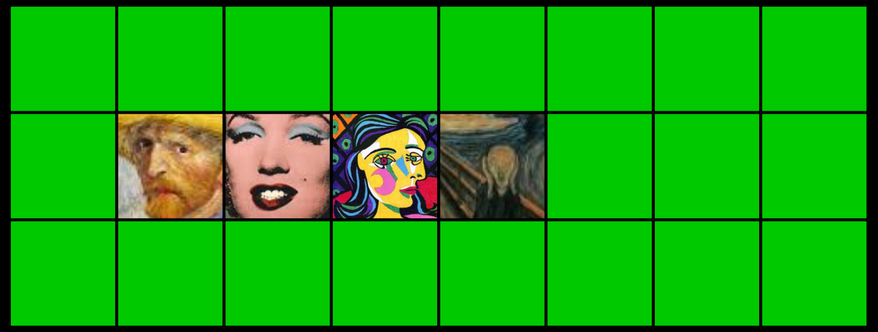


primitive #3: birthday heapspray

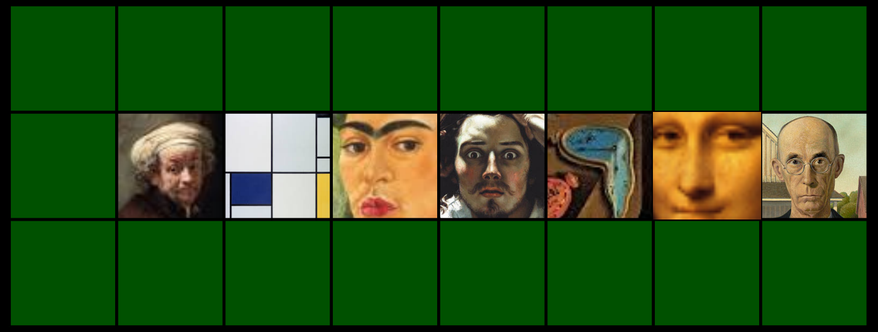
physical memory



attacker memory

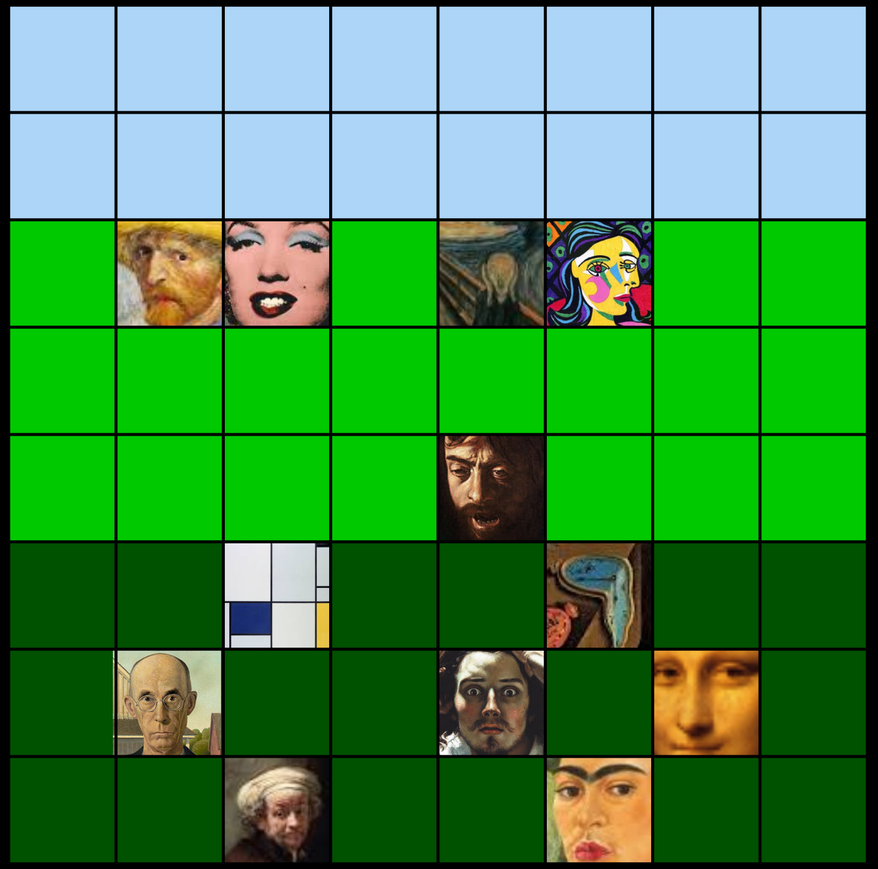


victim memory

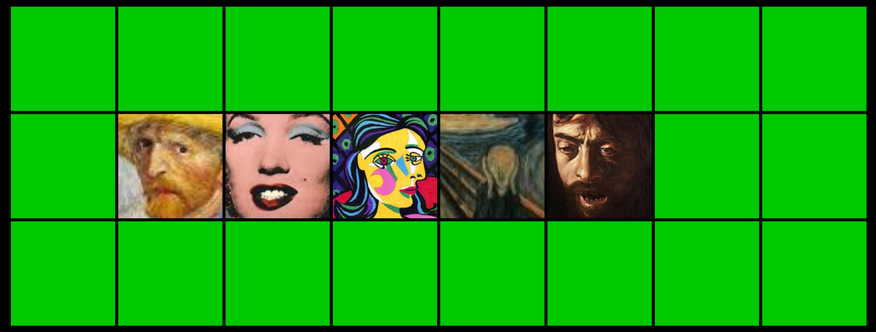


primitive #3: birthday heapspray

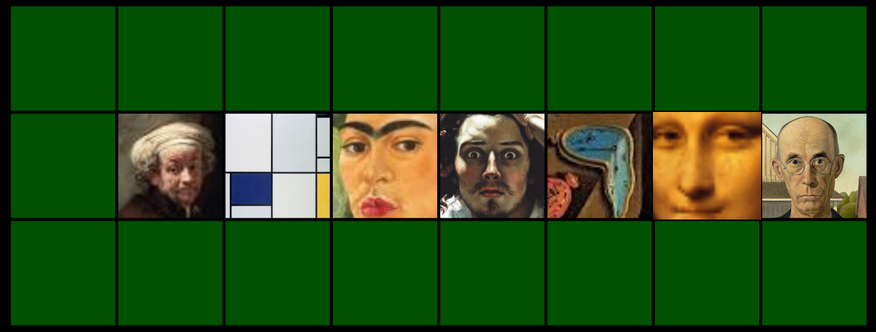
physical memory



attacker memory

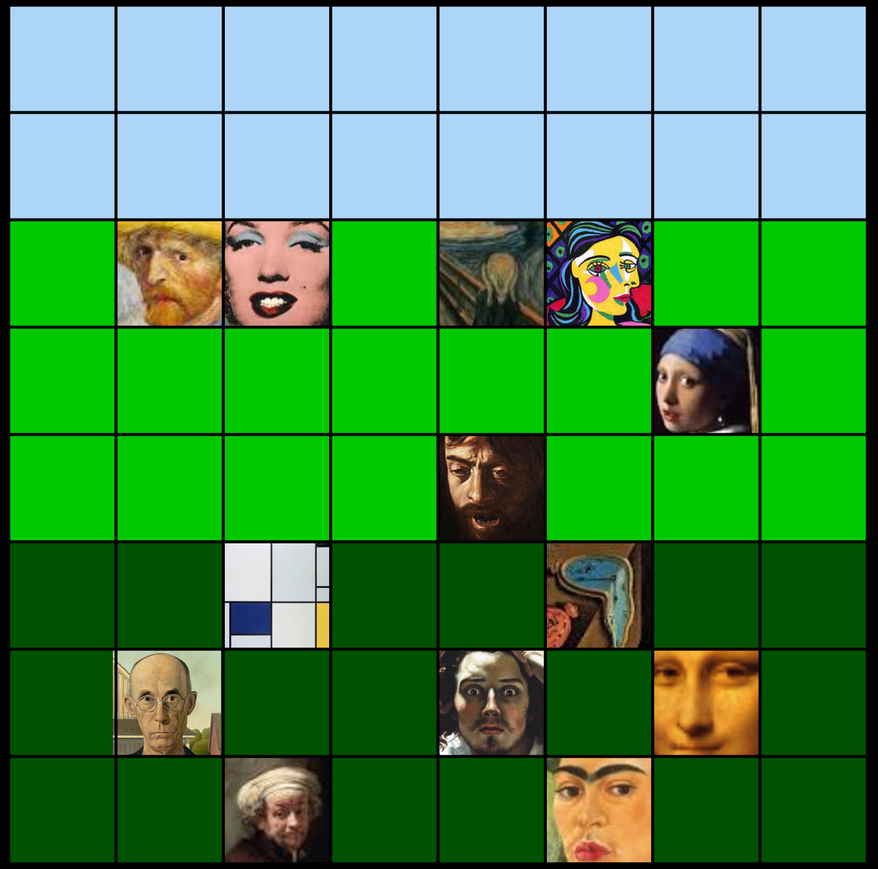


victim memory

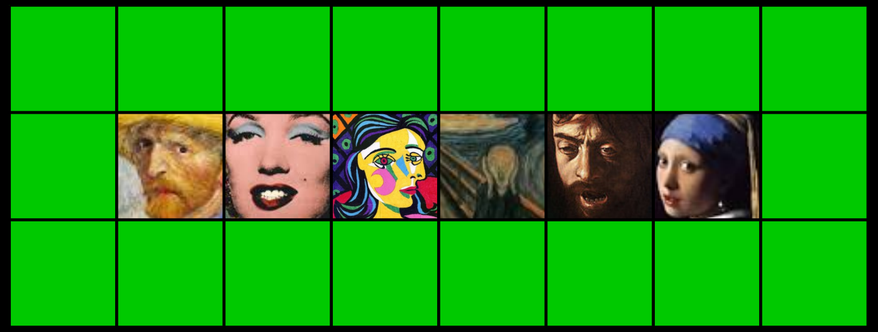


primitive #3: birthday heapspray

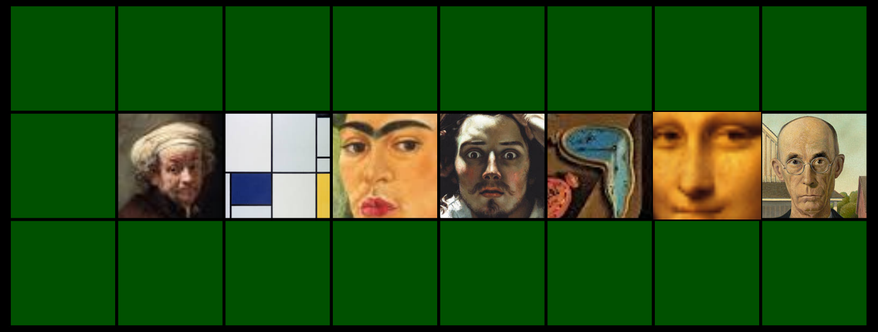
physical memory



attacker memory

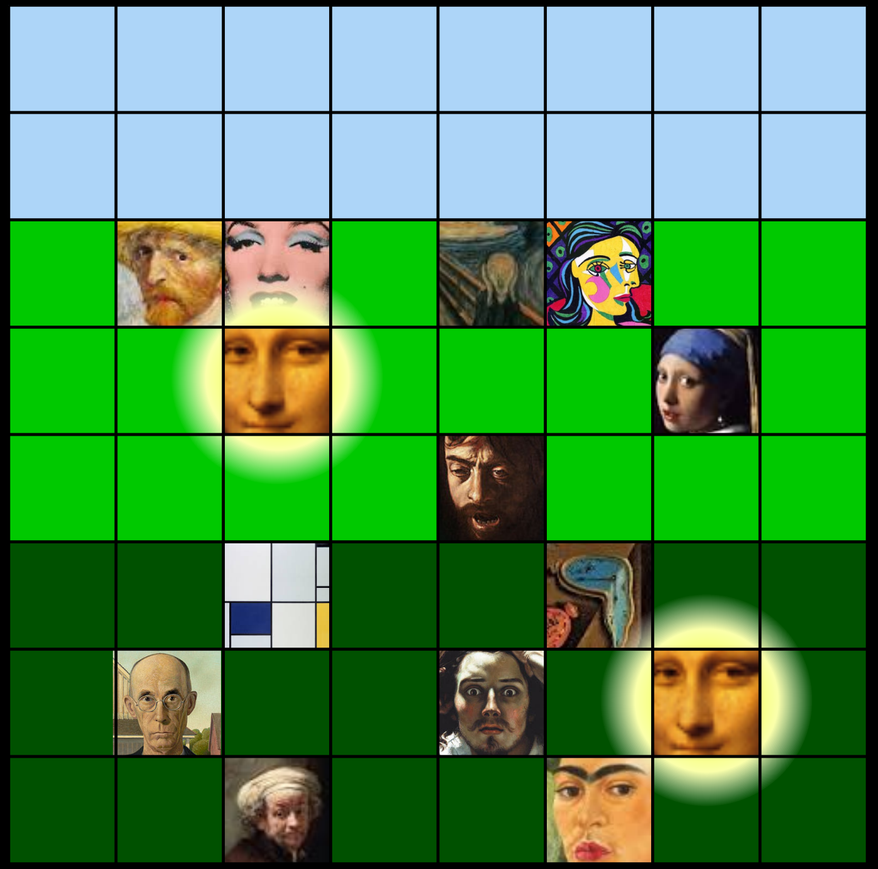


victim memory

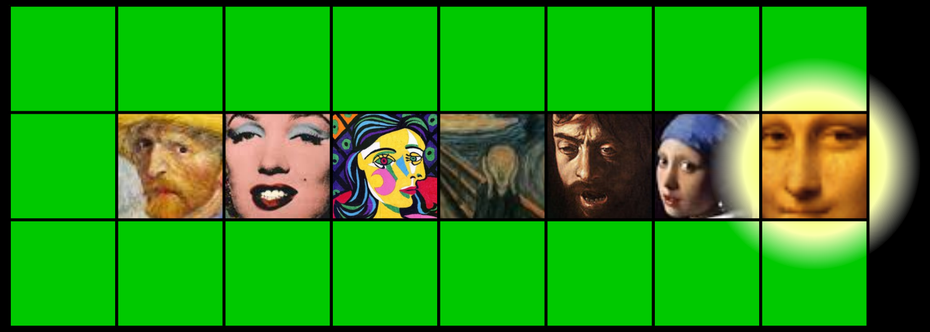


primitive #3: birthday heapspray

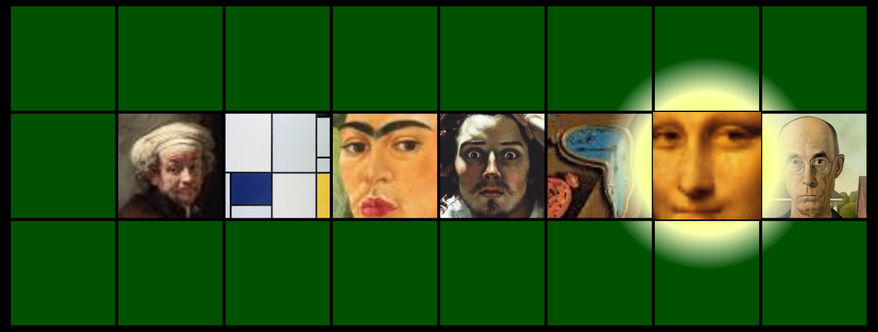
physical memory



attacker memory



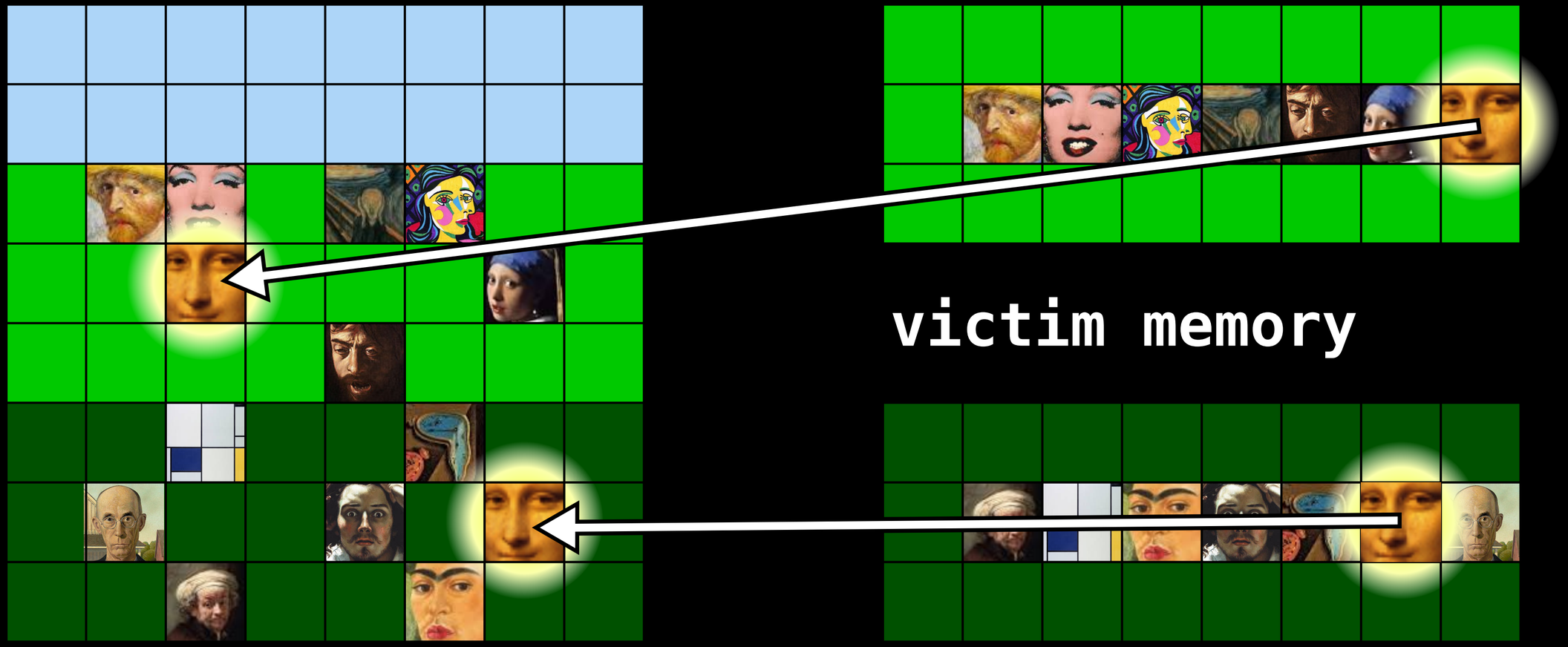
victim memory



primitive #3: birthday heapspray

physical memory

attacker memory



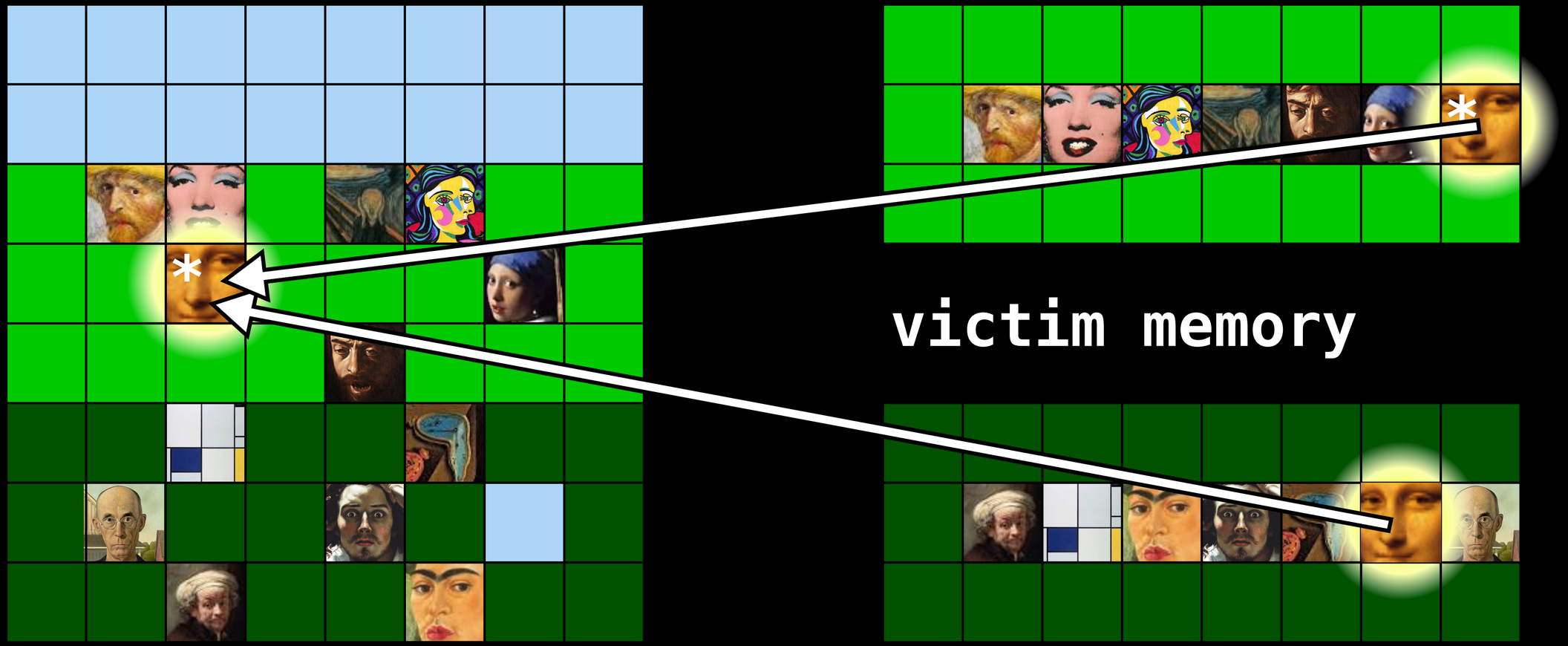
victim memory

primitive #3: birthday heapspray

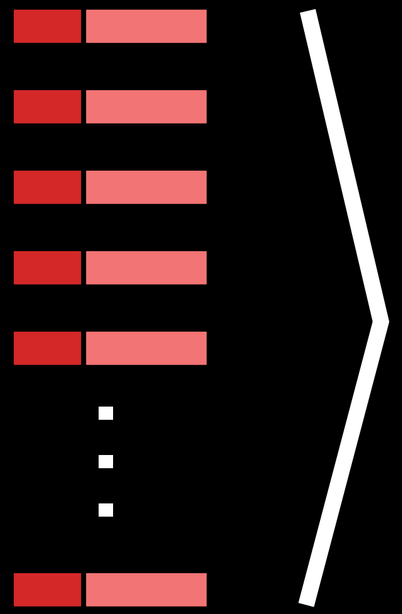
physical memory

attacker memory

victim memory

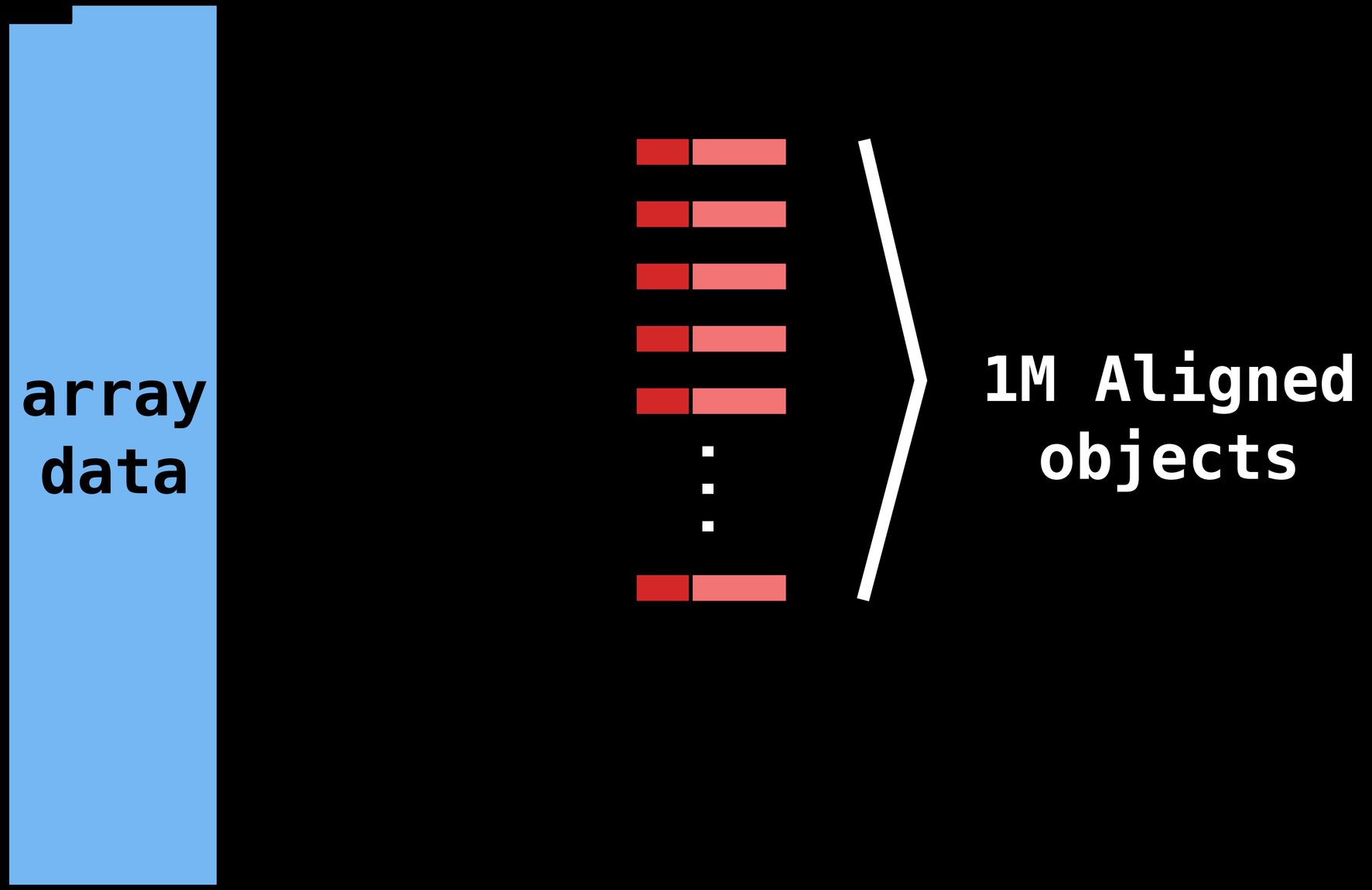


Creating Secret Pages

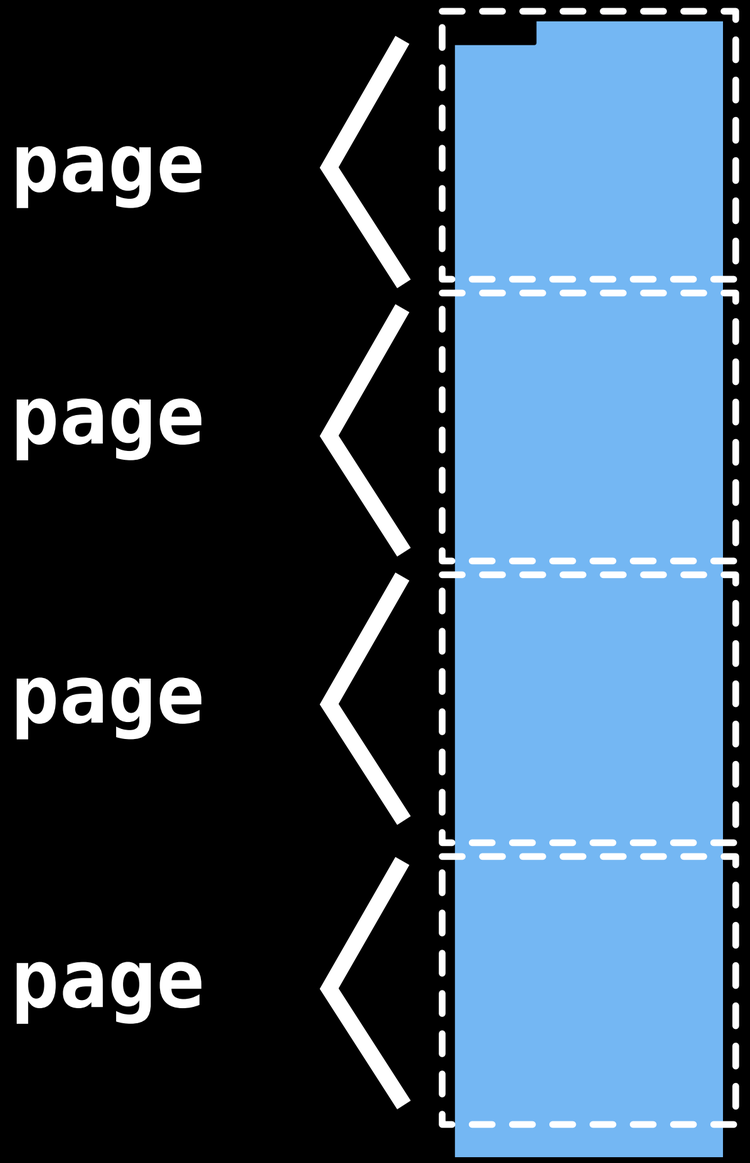


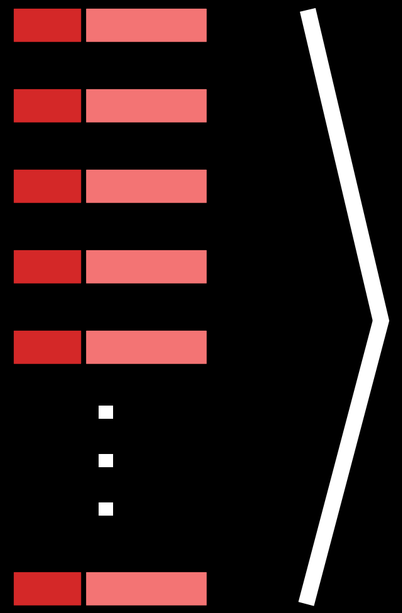
1M Aligned objects

Creating Secret Pages



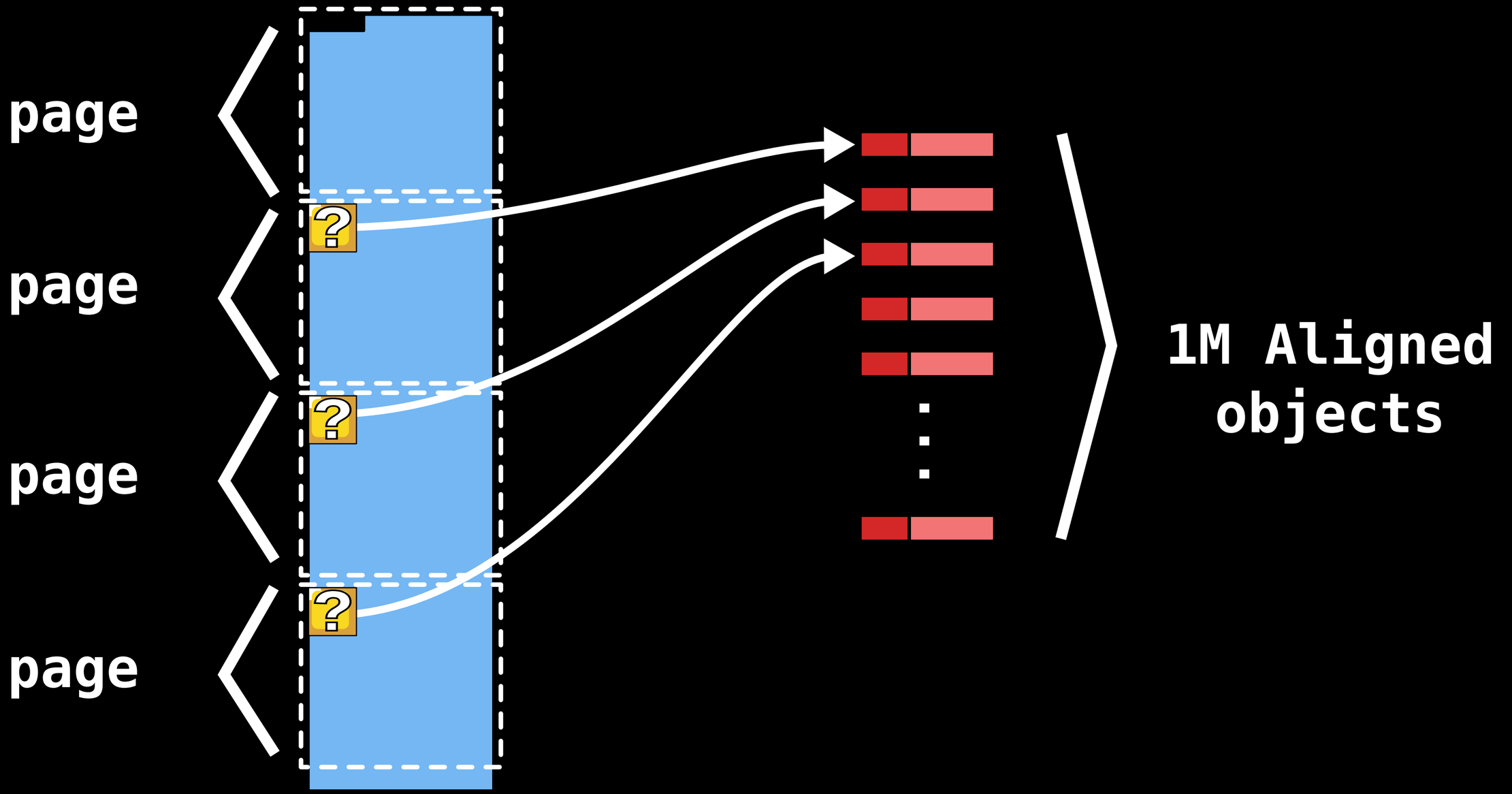
Creating Secret Pages



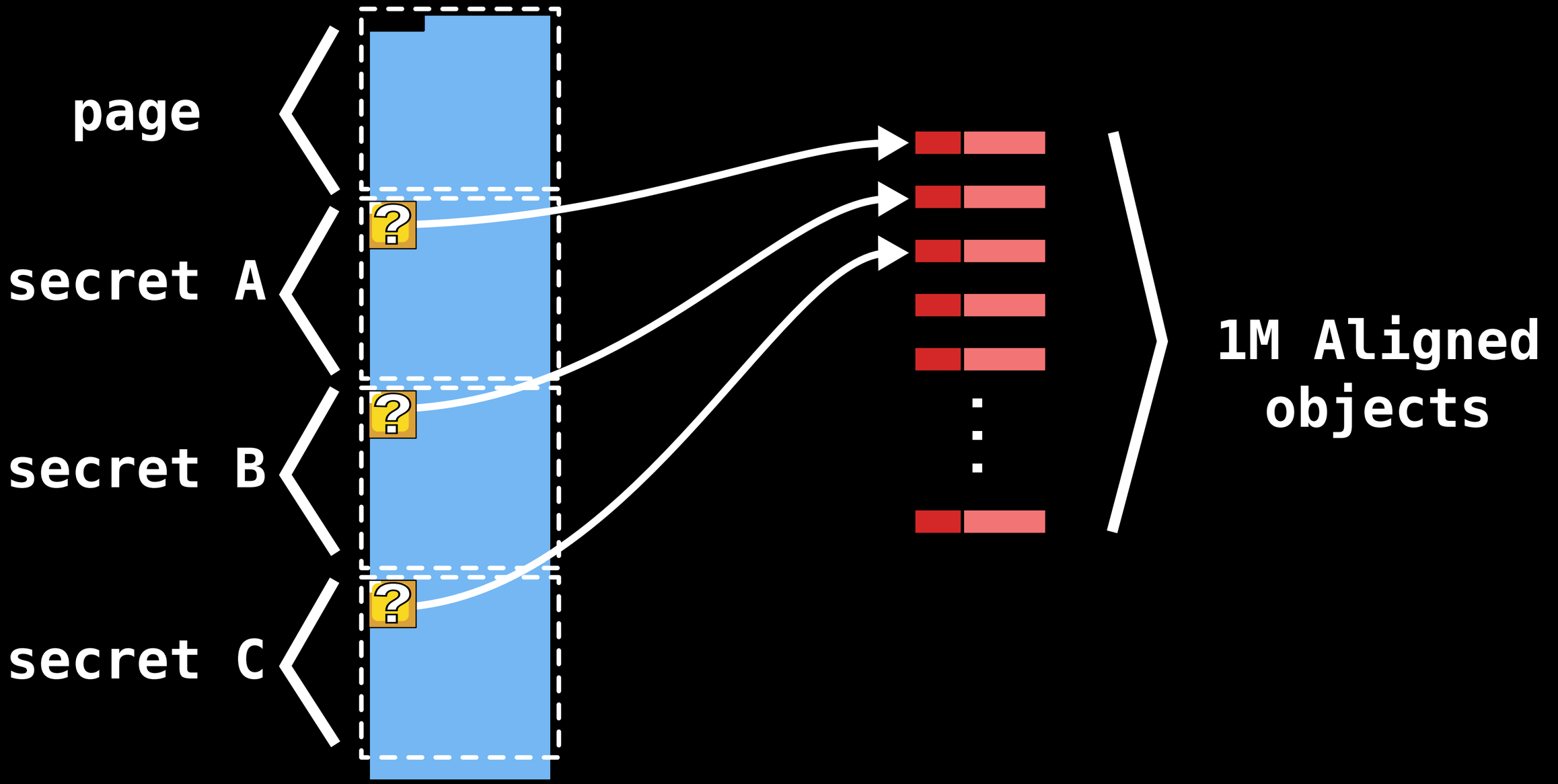


1M Aligned objects

Creating Secret Pages



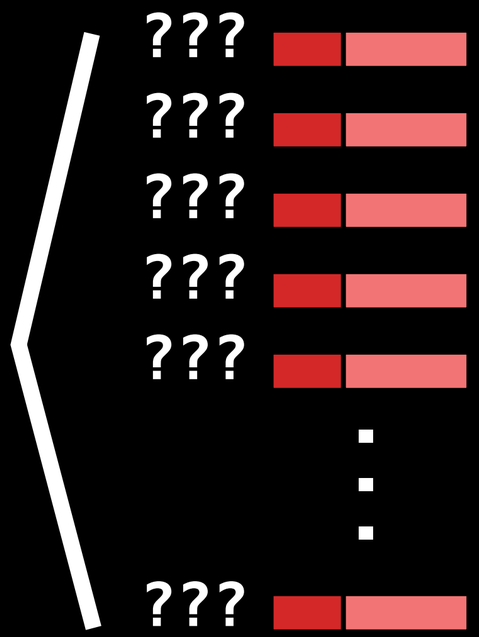
Creating Secret Pages



Creating Probe Pages

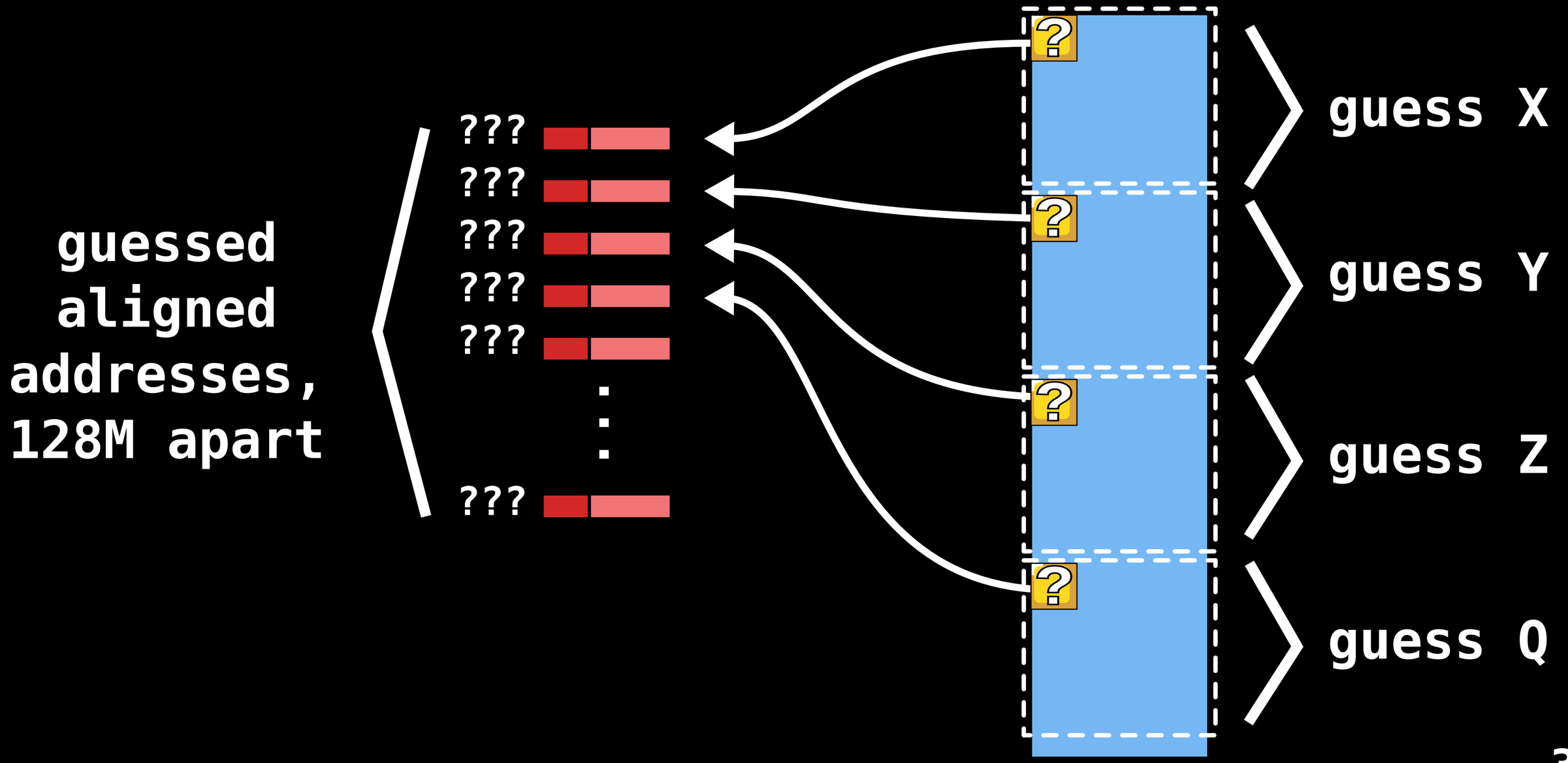
Creating Probe Pages

guessed aligned addresses, 128M apart



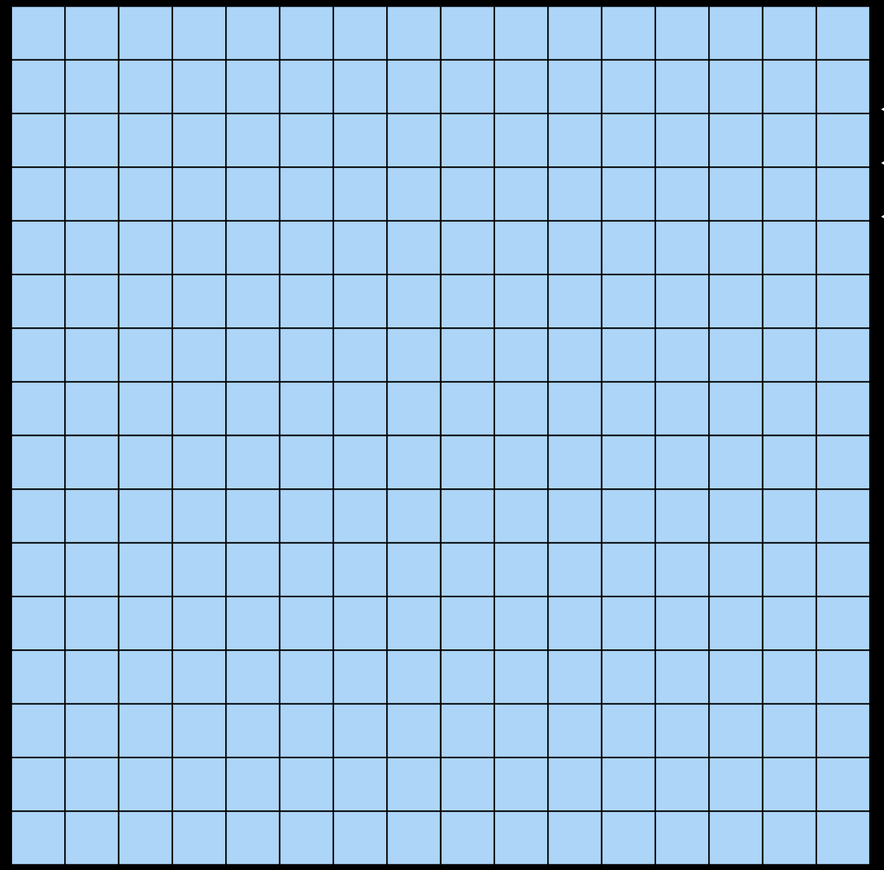
typed array data

Creating Probe Pages



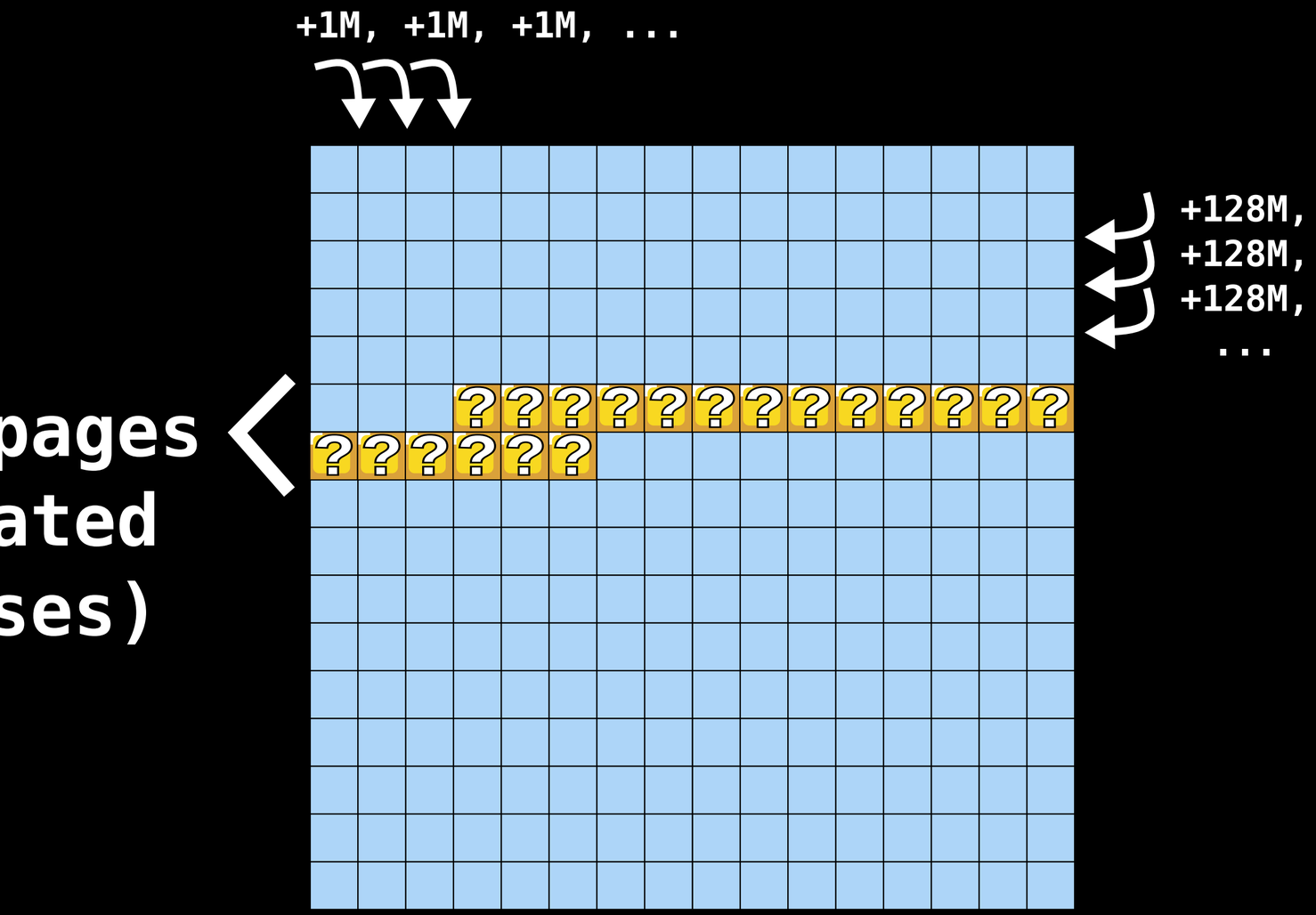
Birthday heap spray

+1M, +1M, +1M, ...

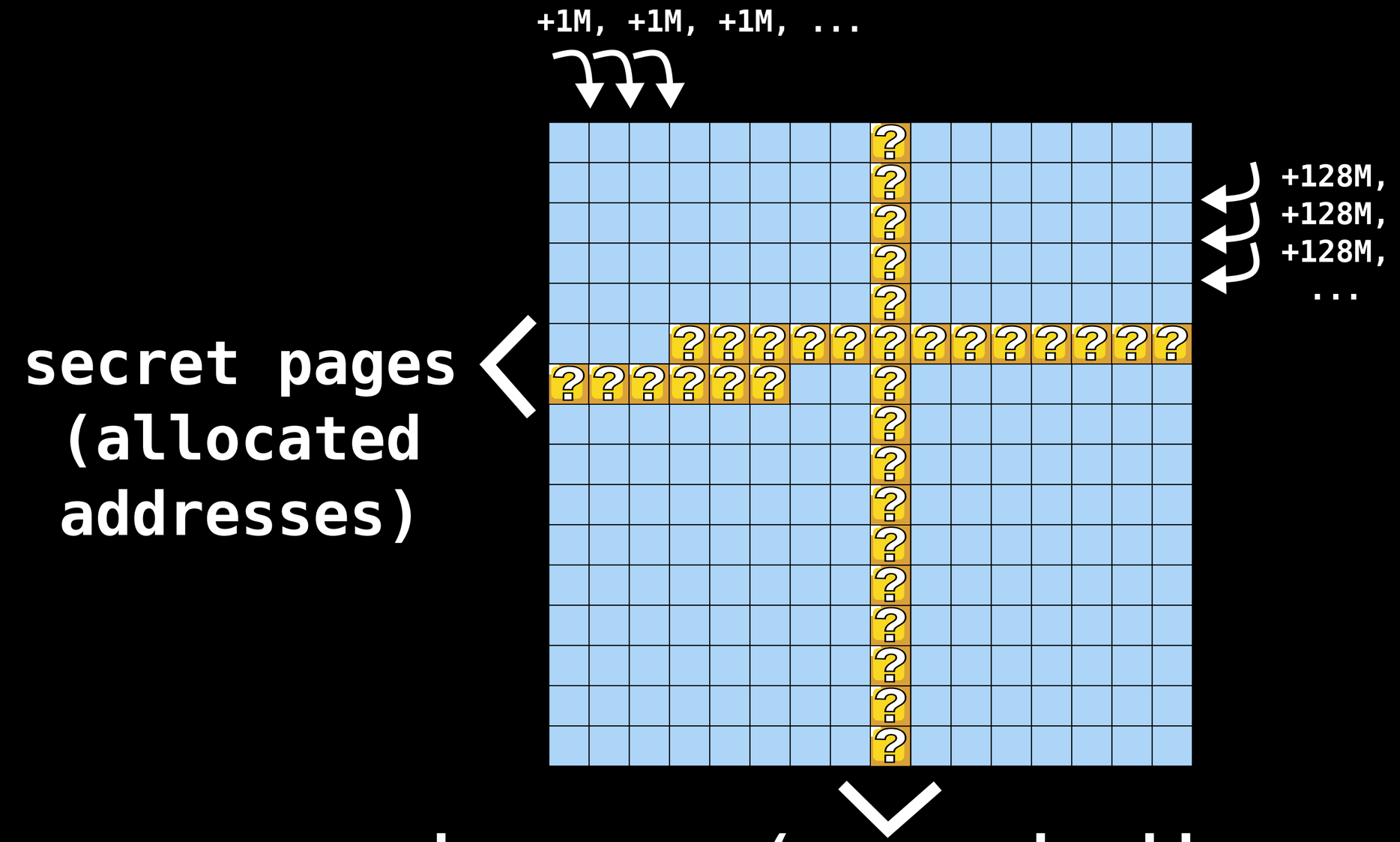


+128M,
+128M,
+128M,
...

Birthday heap spray

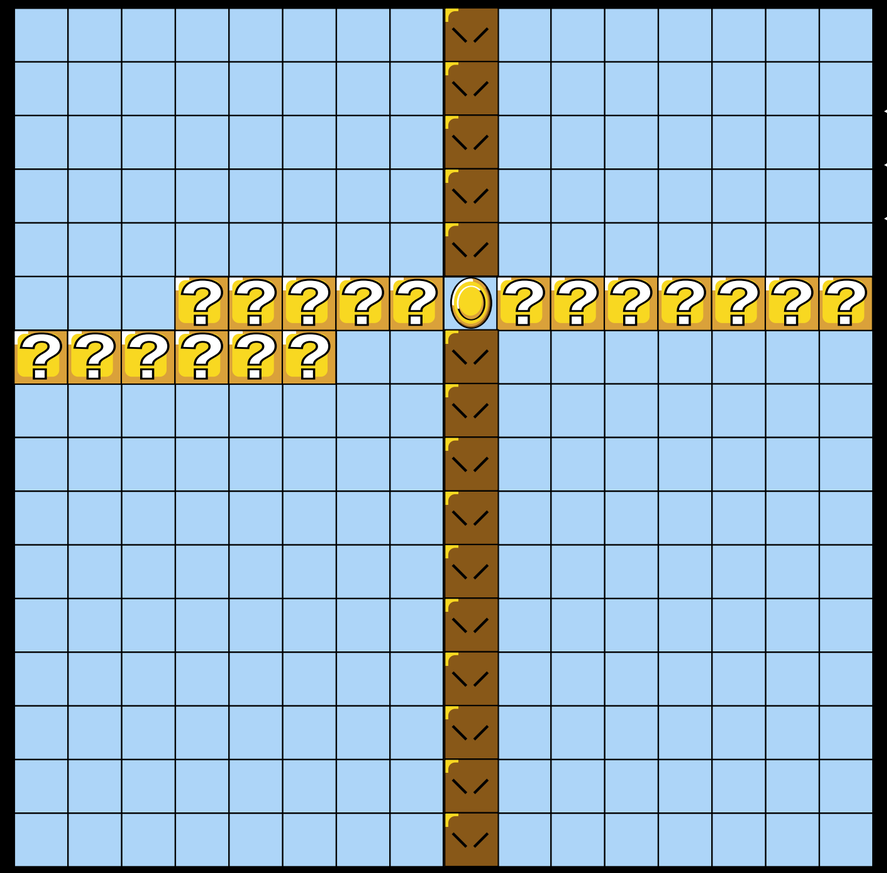


Birthday heap spray



Birthday heap spray

+1M, +1M, +1M, ...



+128M,
+128M,
+128M,
...

secret pages (allocated addresses)

probe pages (guessed addresses)

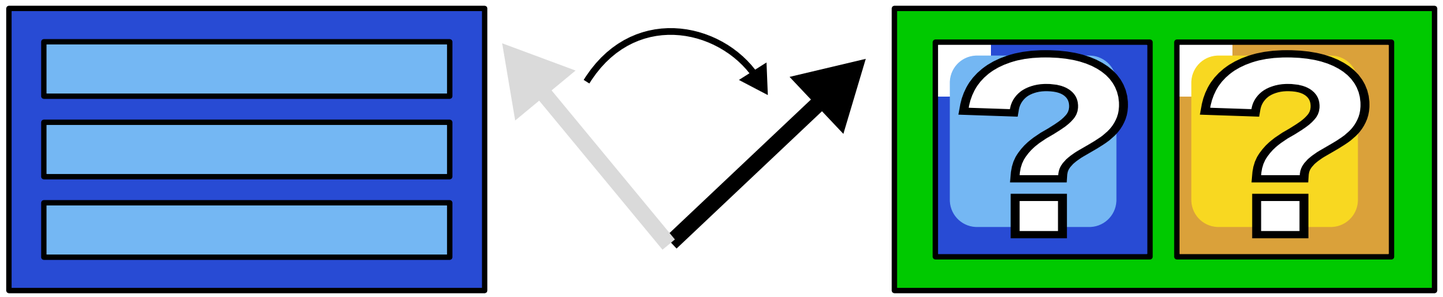
Outline:

Deduplication

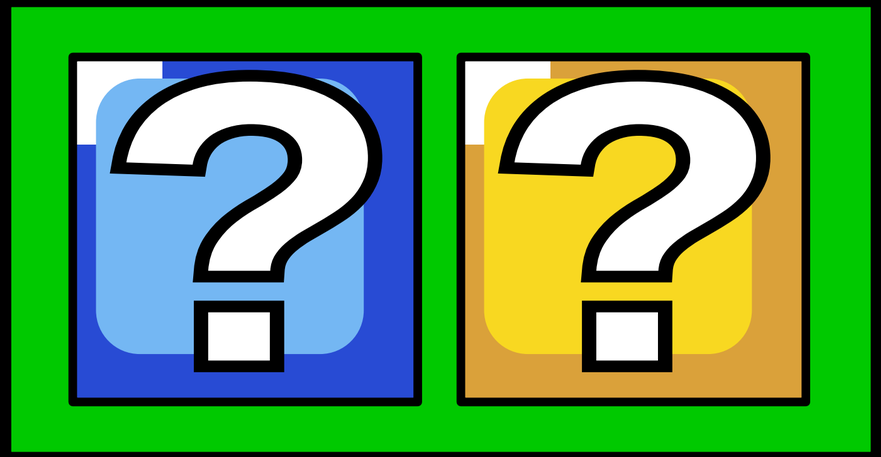
- leak heap & code addresses
- create a fake object

Rowhammer

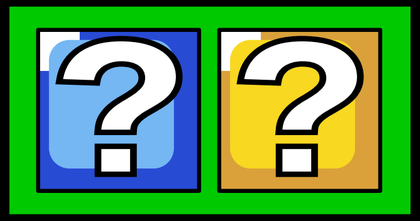
- create reference to our fake object



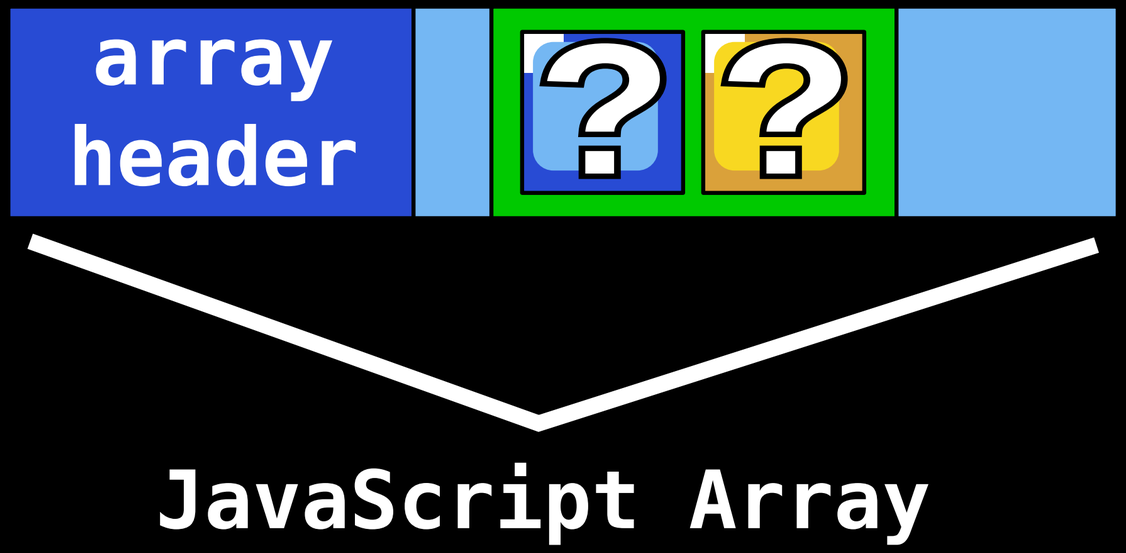
fake Uint8Array object



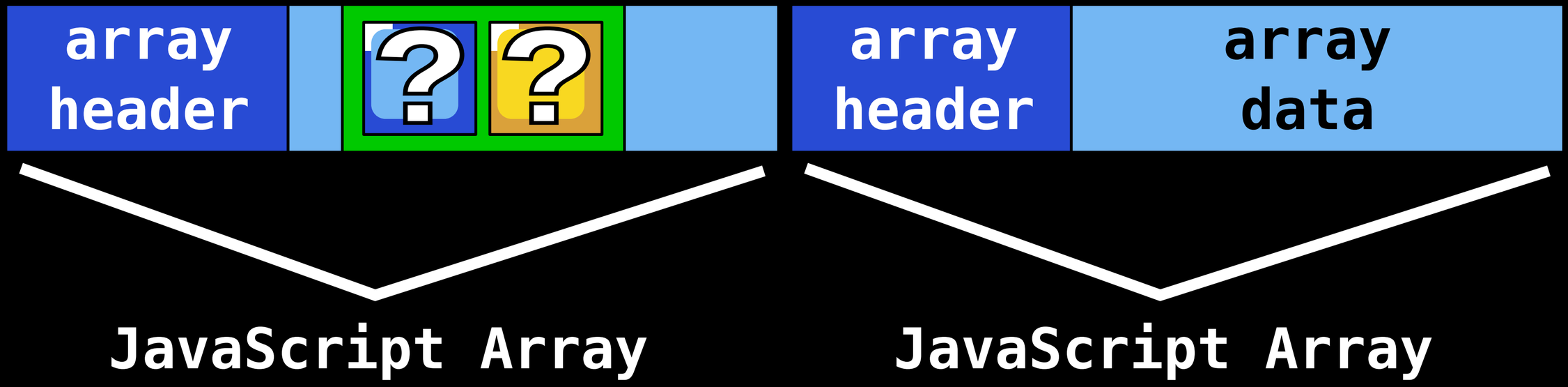
pointer pivotting



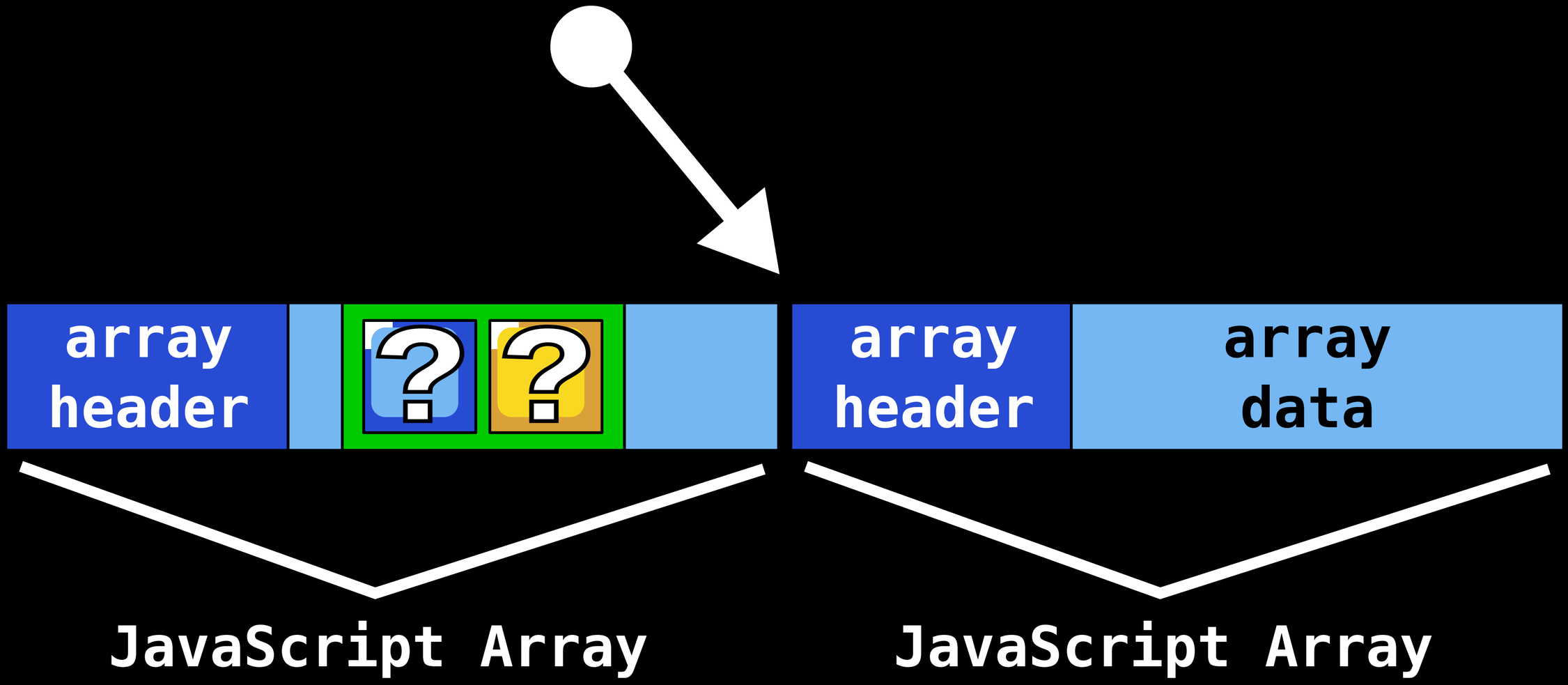
pointer pivotting



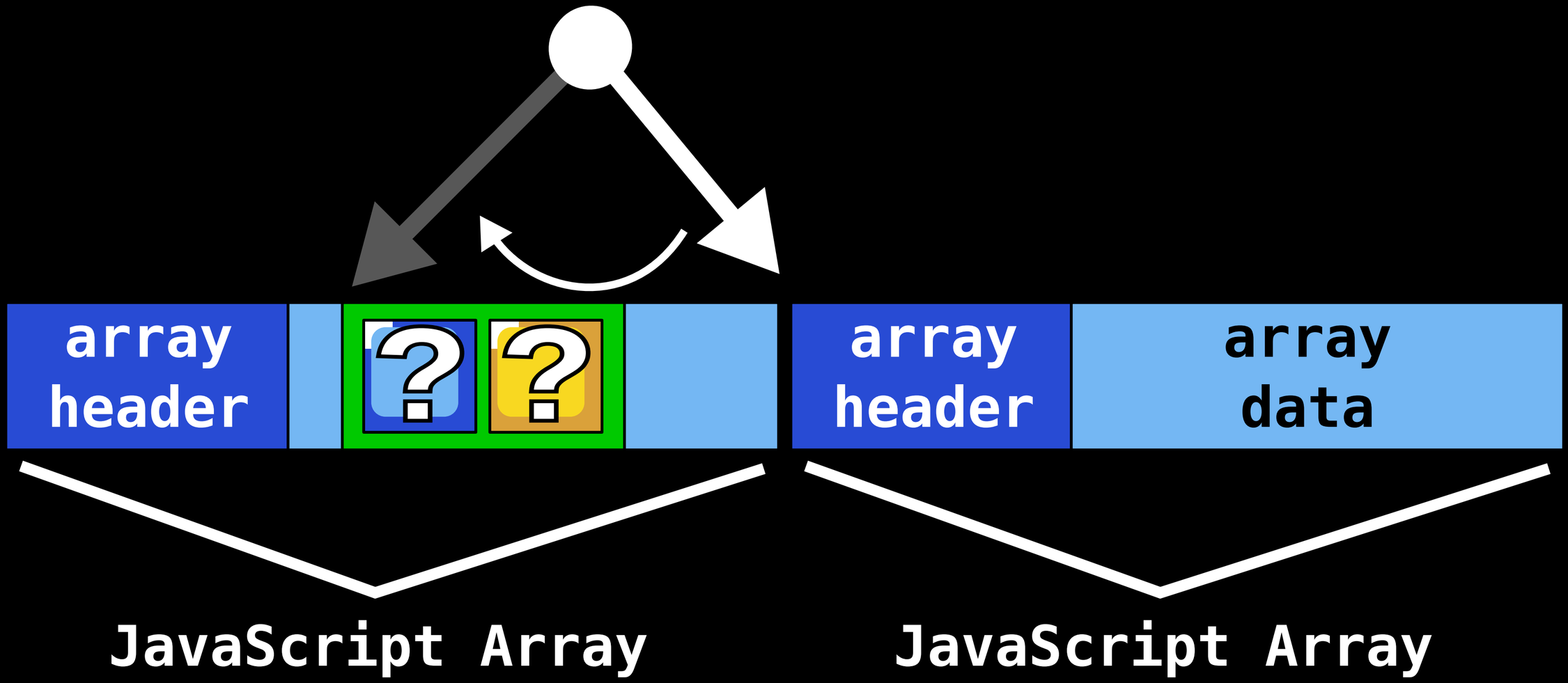
pointer pivotting



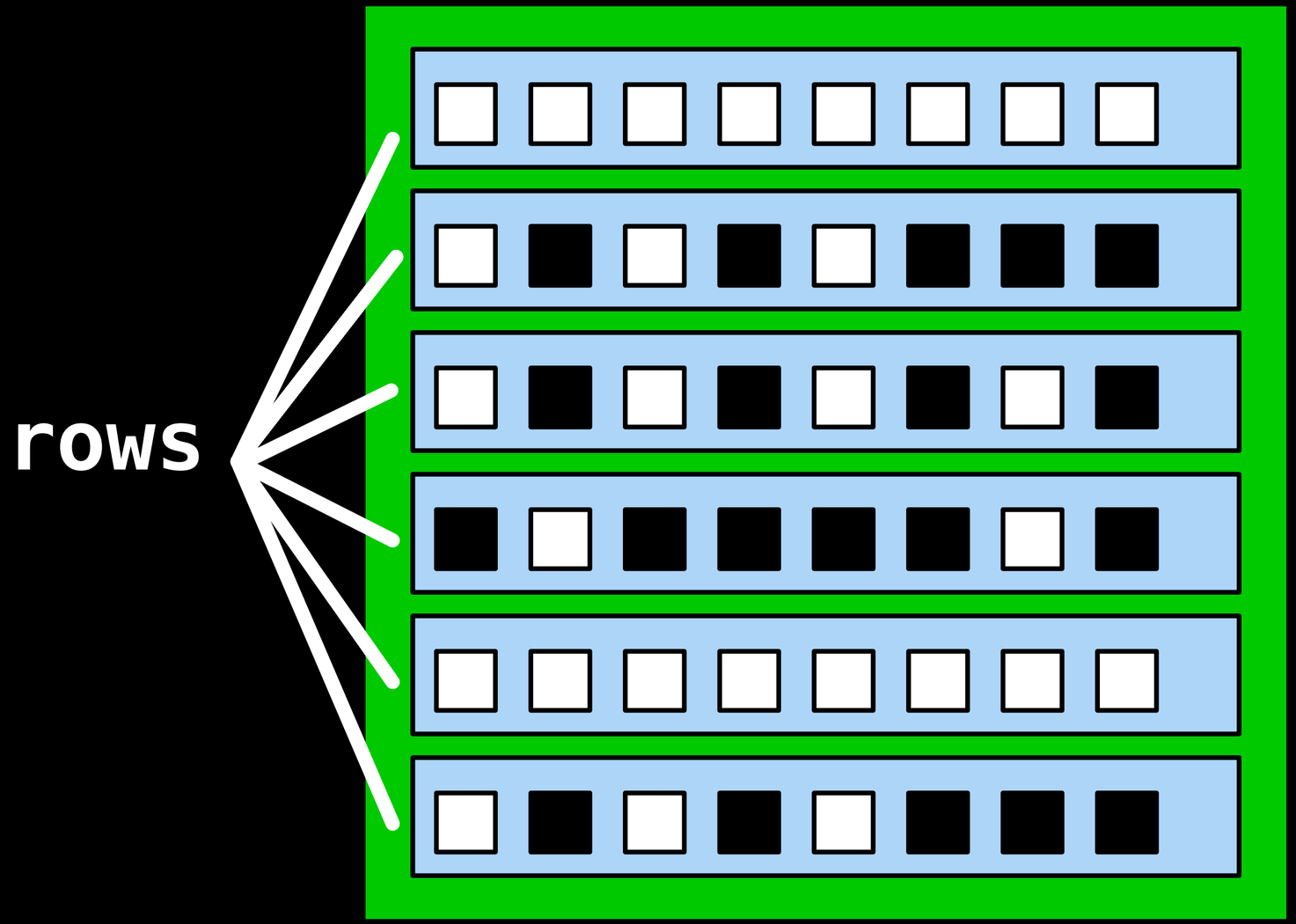
pointer pivotting



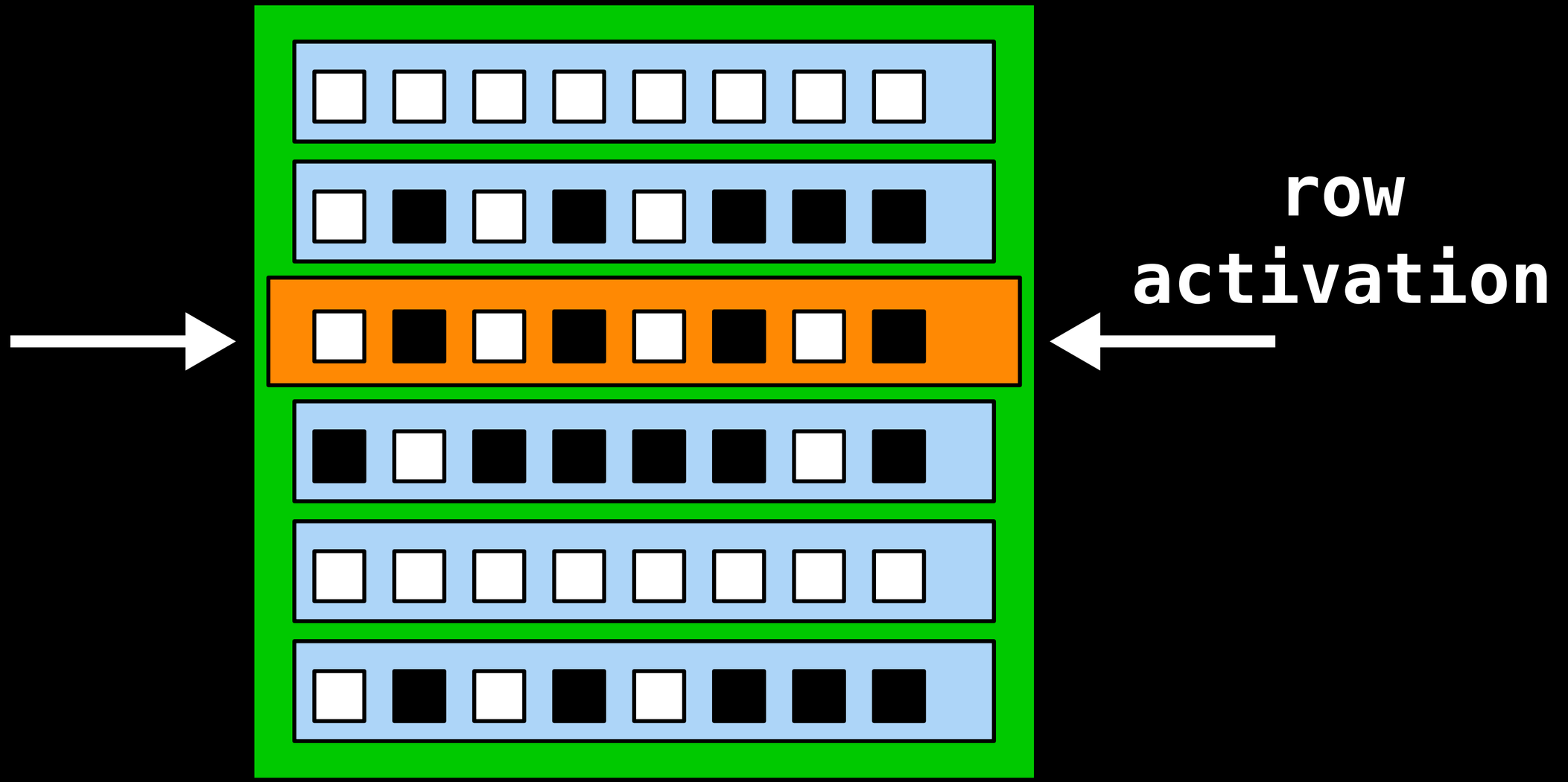
pointer pivotting



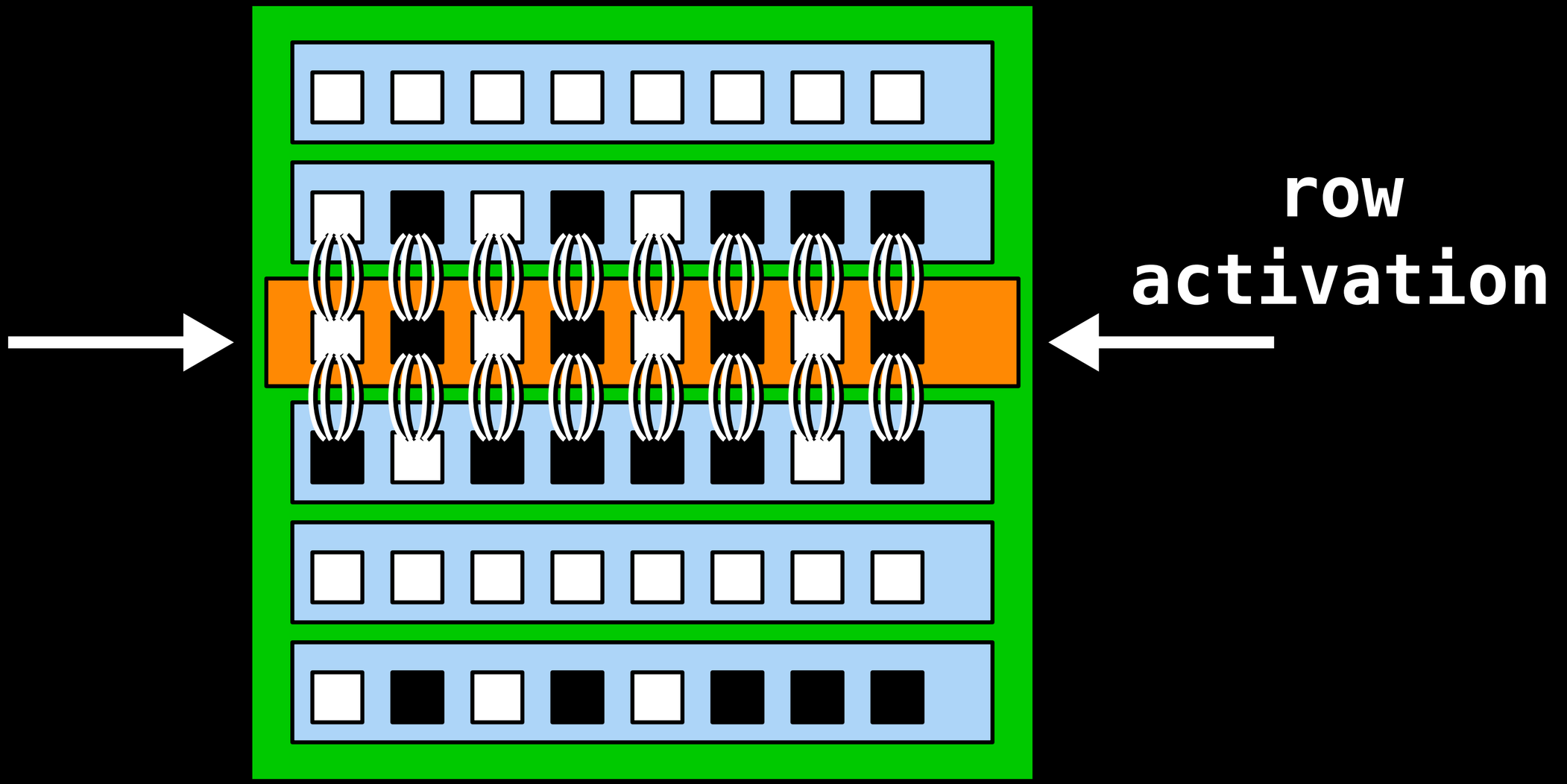
rowhammer attack



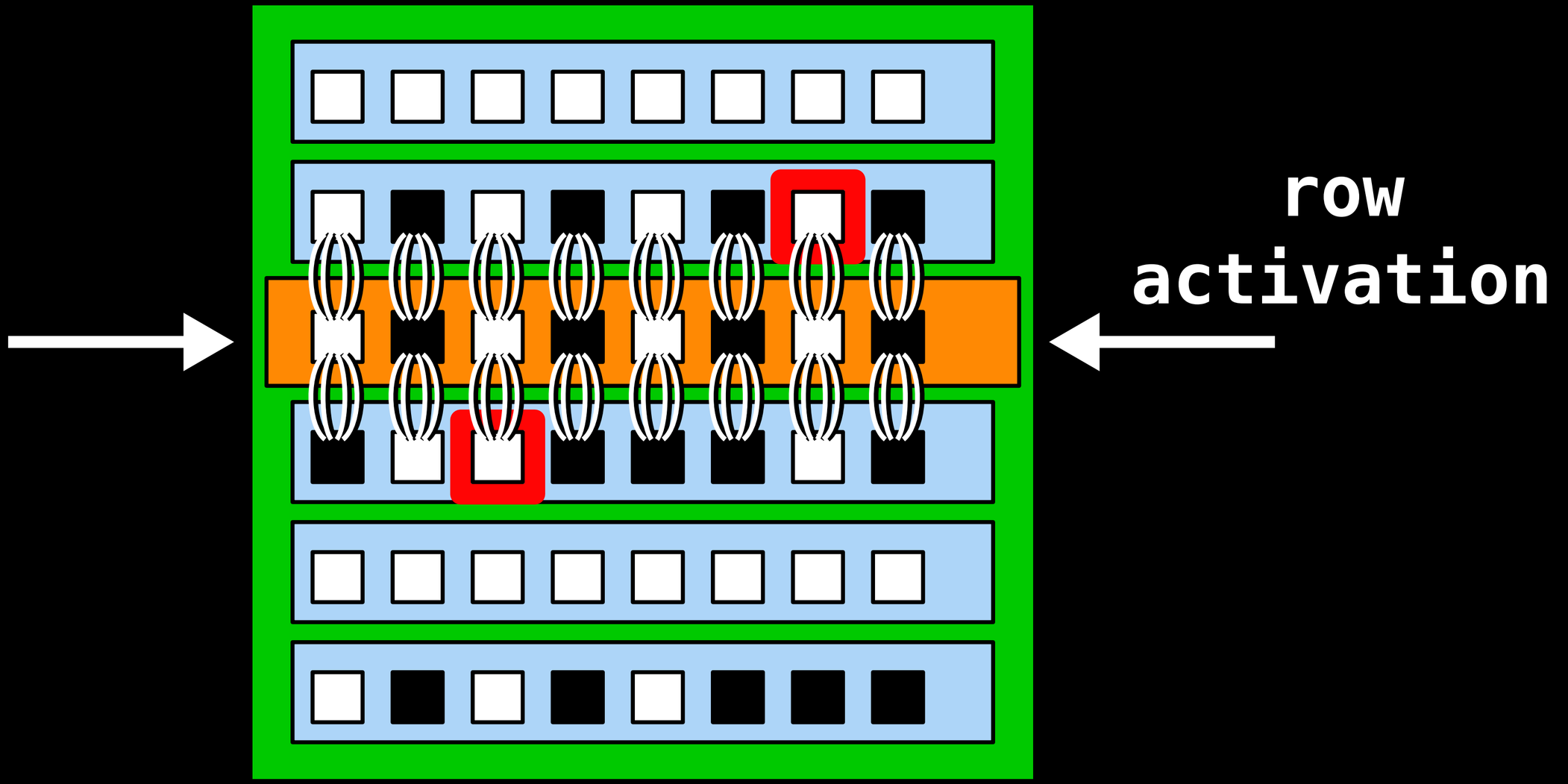
rowhammer attack



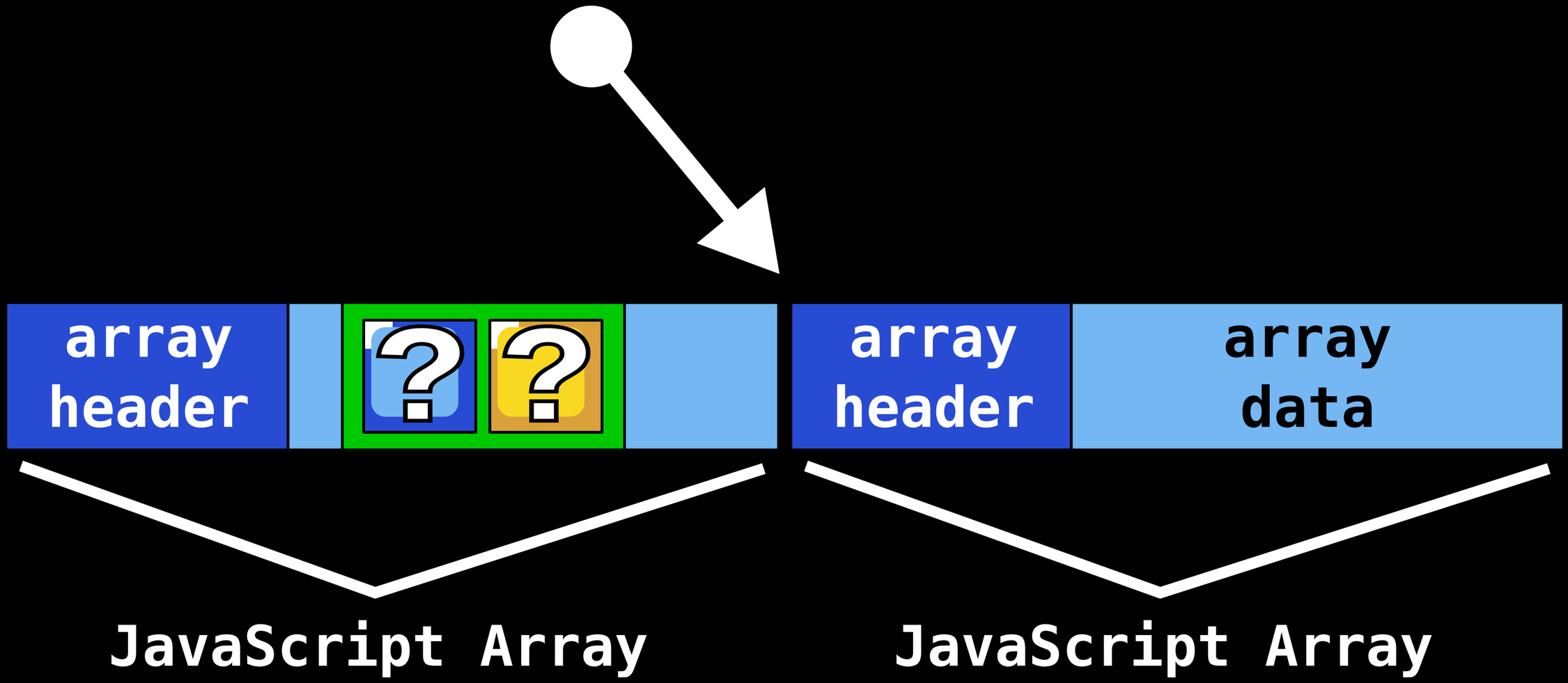
rowhammer attack



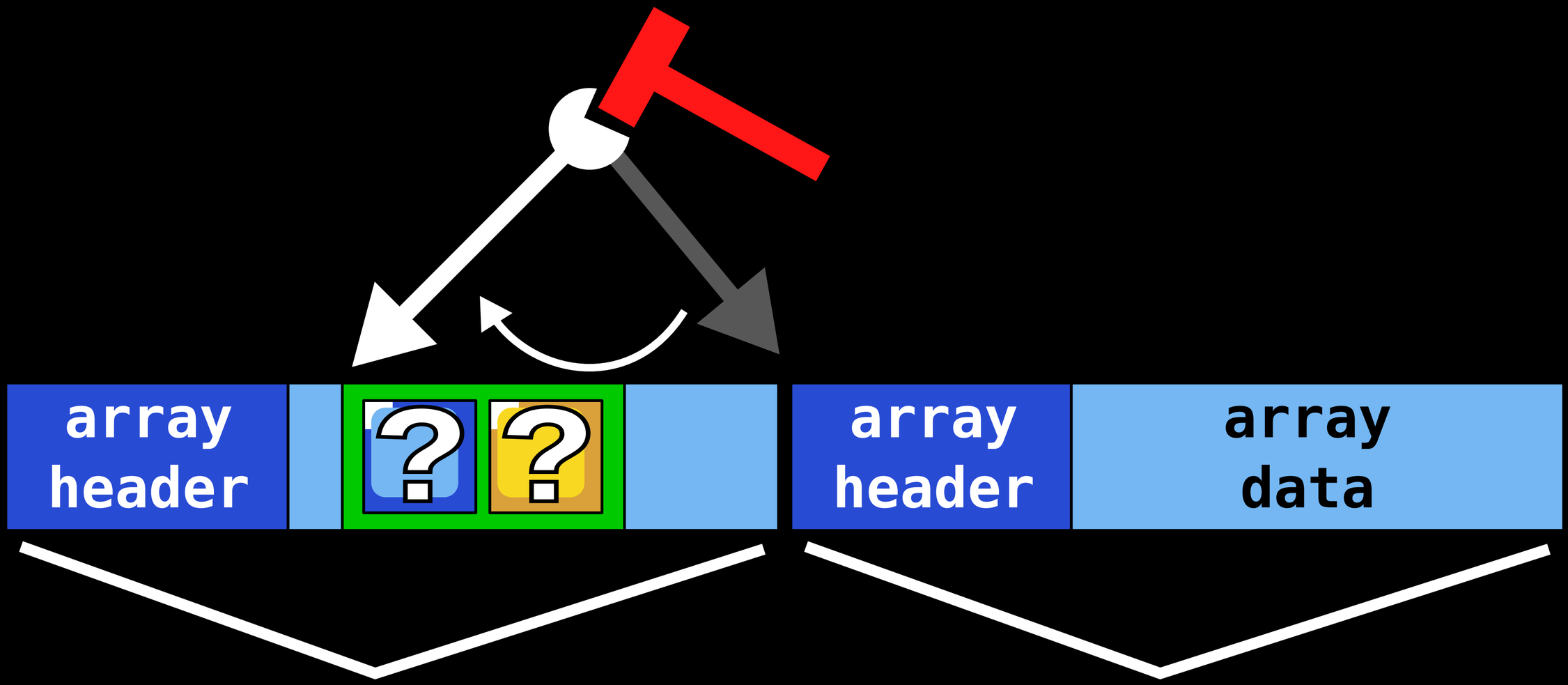
rowhammer attack



pointer pivotting



pointer pivotting



Rowhammer from JavaScript

- Originally: no native bit flips on our DRAM chip (had to lower default refresh rate).

- Now: native bit flips (default settings).

- Had to operate a number of optimizations (e.g., using JS worker threads).

Dedup mitigation

- Disable memory deduplication

```
> Disable-MMAgent -PageCombining
```

- We've reported this issue to Microsoft and they have addressed this issue in ms-16-093, July 18th (CVE-2016-3272) by disabling dedup.

takeaways:

VU VRIJE UNIVERSITEIT AMSTERDAM

- **Dedup Est Machina**: Memory deduplication is a weird machine, and a more powerful side-channel than previously thought.

- Memory saving optimisations, both in hardware and in software come at a price.

- Even without bugs, reliable browser exploitation in JavaScript is possible, using dedup+rowhammer.

https://www.vusec.net/projects/dedup-est-machina