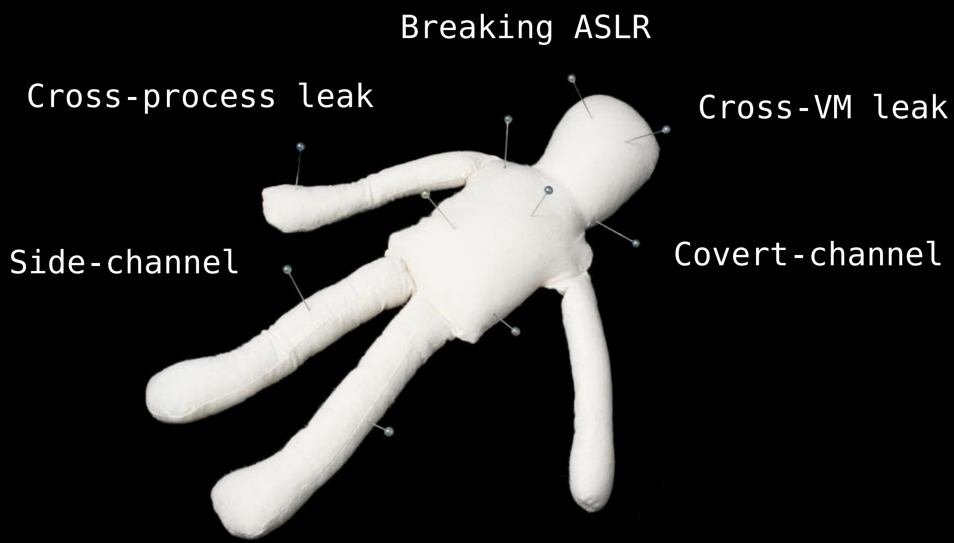
Memory Deduplication: The Curse that Keeps on Giving



VU VRIJE UNIVERSITEIT AMSTERDAM

Erik Bosman, Ben Gras, Kaveh Razavi

xorlab

Antonio Barresi

HELLO, THIS IS 33c3 WORKS FOR ME
27.-30.12.2016 | CCH HAMBURG | GERMANY

Who we are

Erik Bosman

erik@minemu.org

@brainsmoke

PhD candidate at

Research on building reliable and secure computing systems.

https://www.vusec.net/

Antonio Barresi

antonio.barresi@xorlab.com

@AntonioHBarresi

Co-founder of

Interested in software and systems security topics.

https://www.xorlab.com

Acknowledgments

VU VRIJE UNIVERSITEIT AMSTERDAM

Cristiano Giuffrida

Herbert Bos

KATHOLIEKE UNIVERSITEIT LEUVEN

Bart Preneel

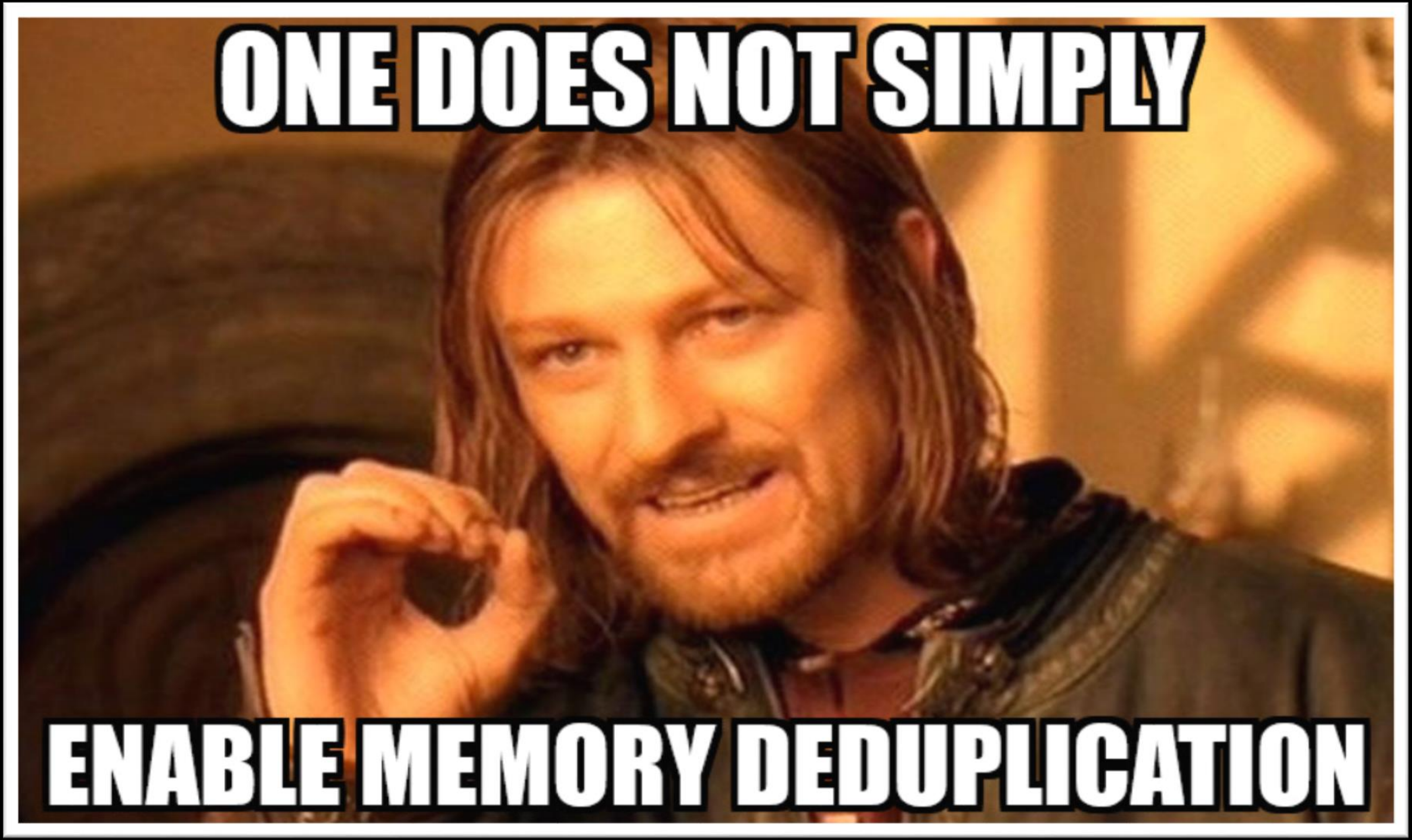
PURDUE UNIVERSITY

Mathias Payer

ETH zürich

Thomas R. Gross

Our message today…


ONE DOES NOT SIMPLY
ENABLE MEMORY DEDUPLICATION

Outline

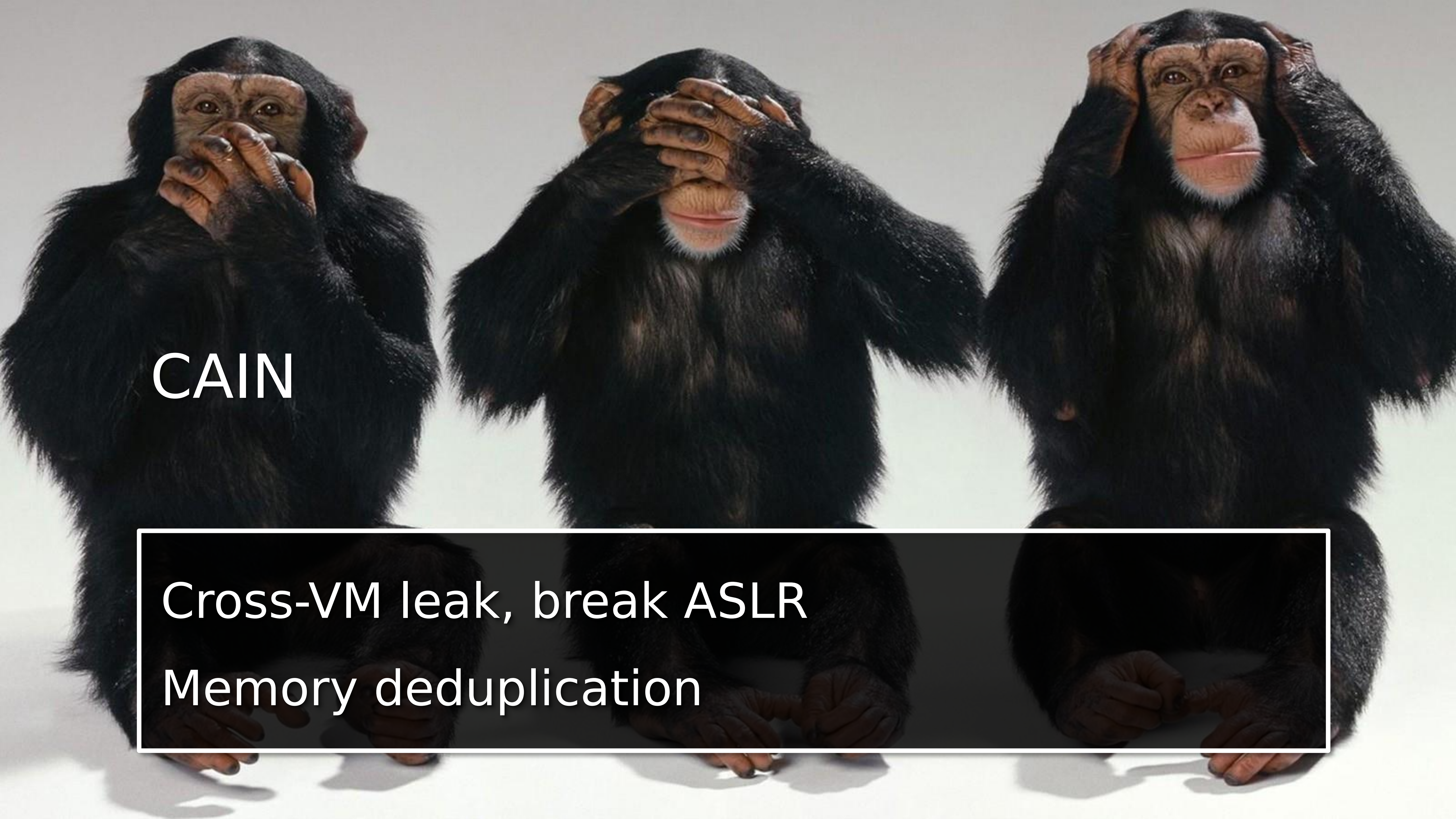
Outline

> Memory deduplication

Outline

> Memory deduplication
> Side-channel

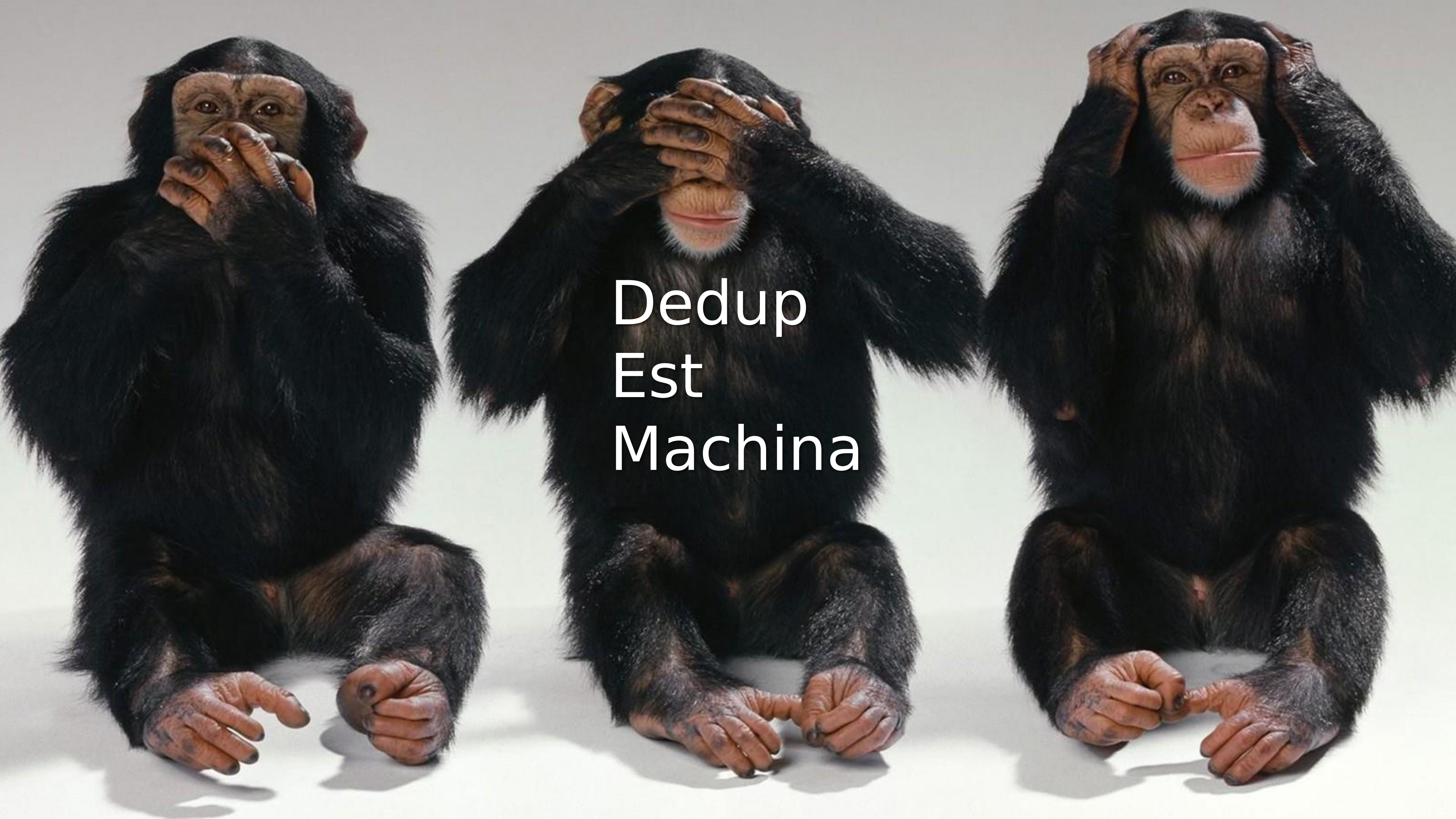

CAIN

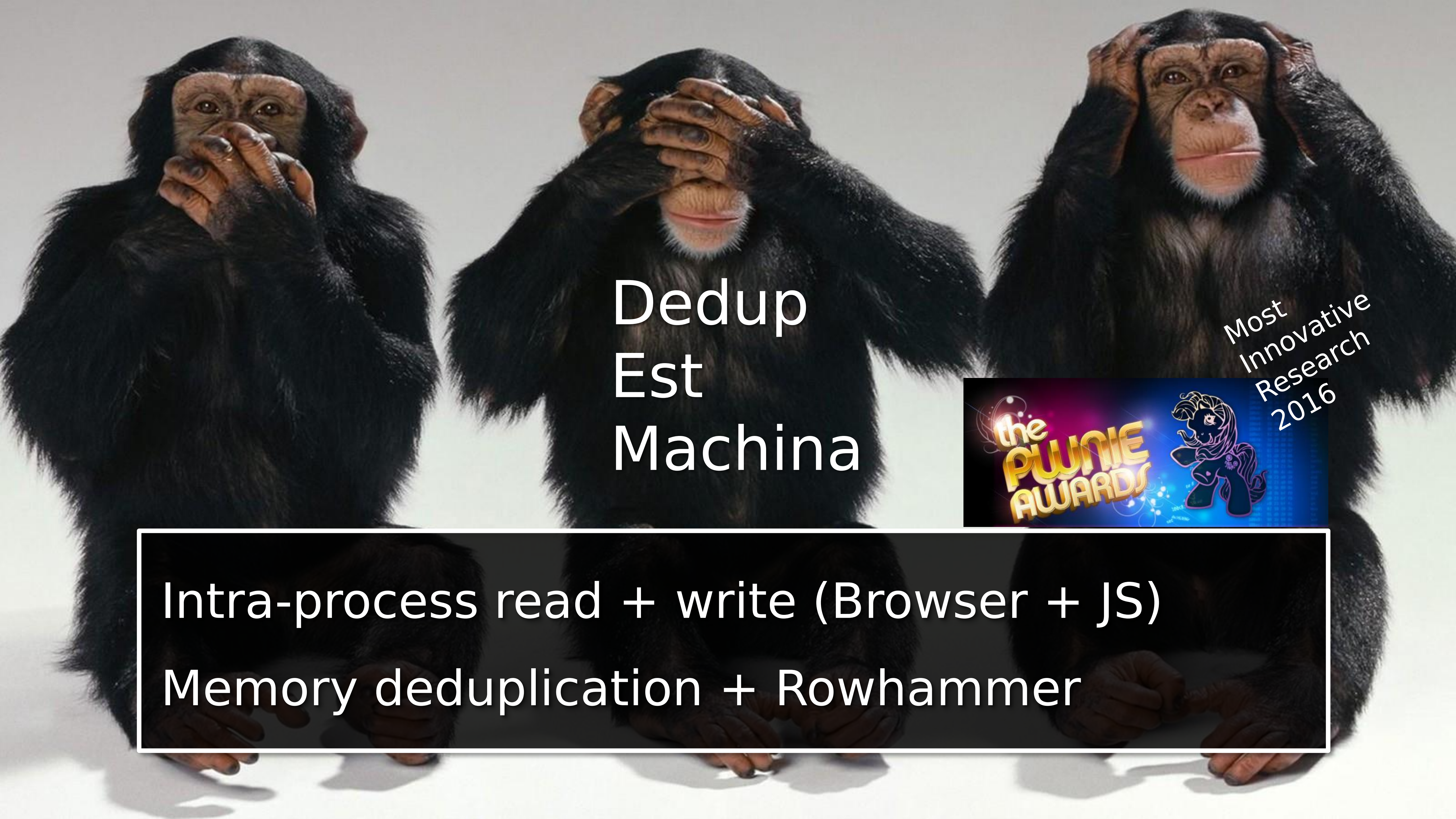


CAIN

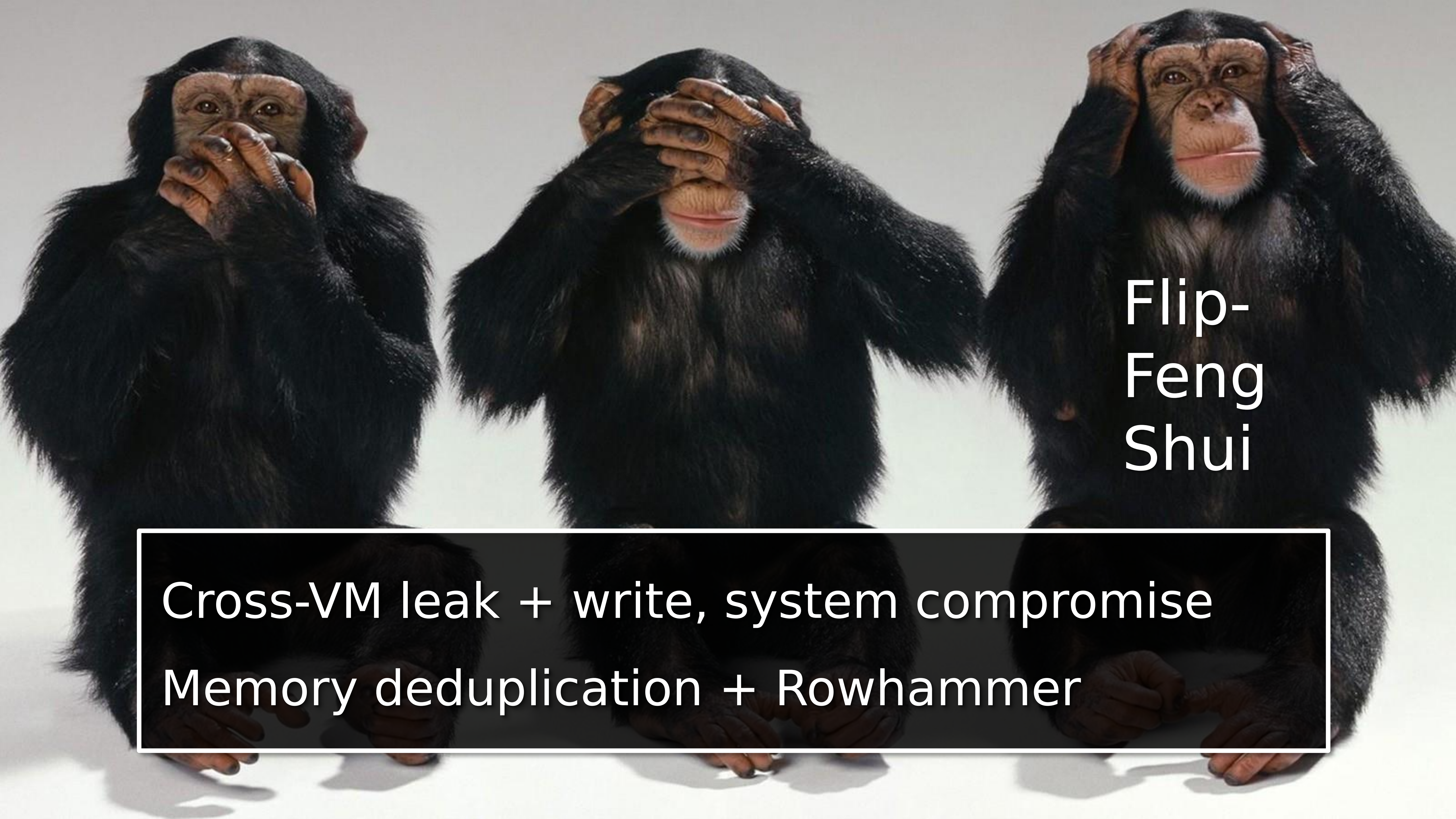
Cross-VM leak, break ASLR

Memory deduplication


Dedup
Est
Machina


Dedup
Est
Machina
Most
Innovative
Research
2016
the PWNIE AWARDS
Intra-process read + write (Browser + JS)
Memory deduplication + Rowhammer


Flip-
Feng
Shui


Flip-
Feng
Shui
Cross-VM leak + write, system compromise
Memory deduplication + Rowhammer

Outline

> Memory deduplication
> Side-channel

> CAIN attack (2015)
> Dedup Est Machina (2016)
> Flip-Feng Shui (2016)

Outline

> Memory deduplication
> Side-channel

> CAIN attack (2015)
> Dedup Est Machina (2016)
> Flip-Feng Shui (2016)

> Conclusion

black hat
USA 2016

Memory deduplication

Memory deduplication

A method of reducing memory usage.

Memory deduplication

A method of reducing memory usage.

Used in virtualization environments,

Memory deduplication

A method of reducing memory usage.

Used in virtualization environments,

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

Memory deduplication

In virtualized environments it allows to reclaim memory and supports overcommitment of memory.

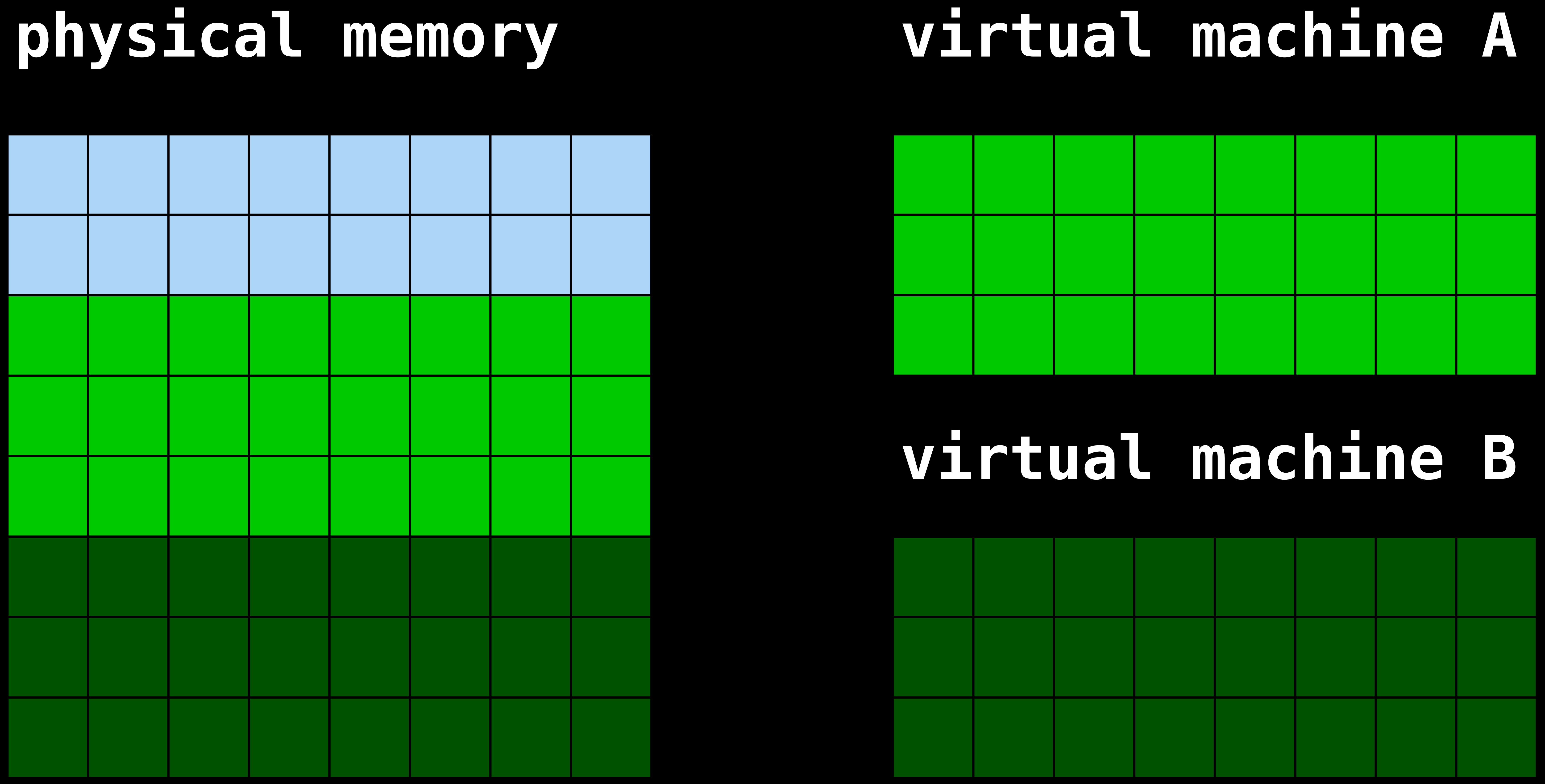
Memory deduplication

In virtualized environments it allows to reclaim memory and supports overcommitment of memory.

= run more VMs

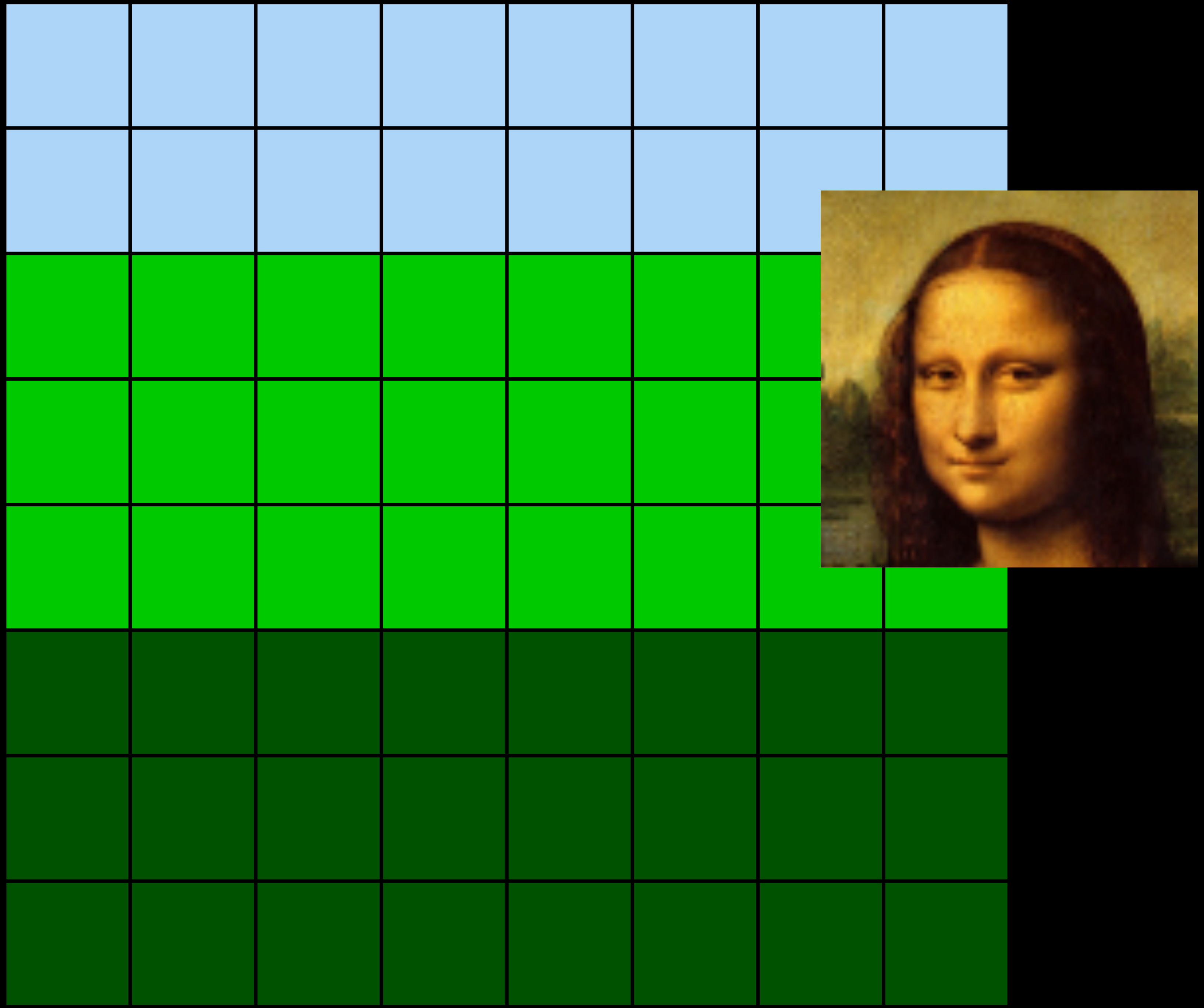

Now we can sell even more VMs... $$$

Memory deduplication

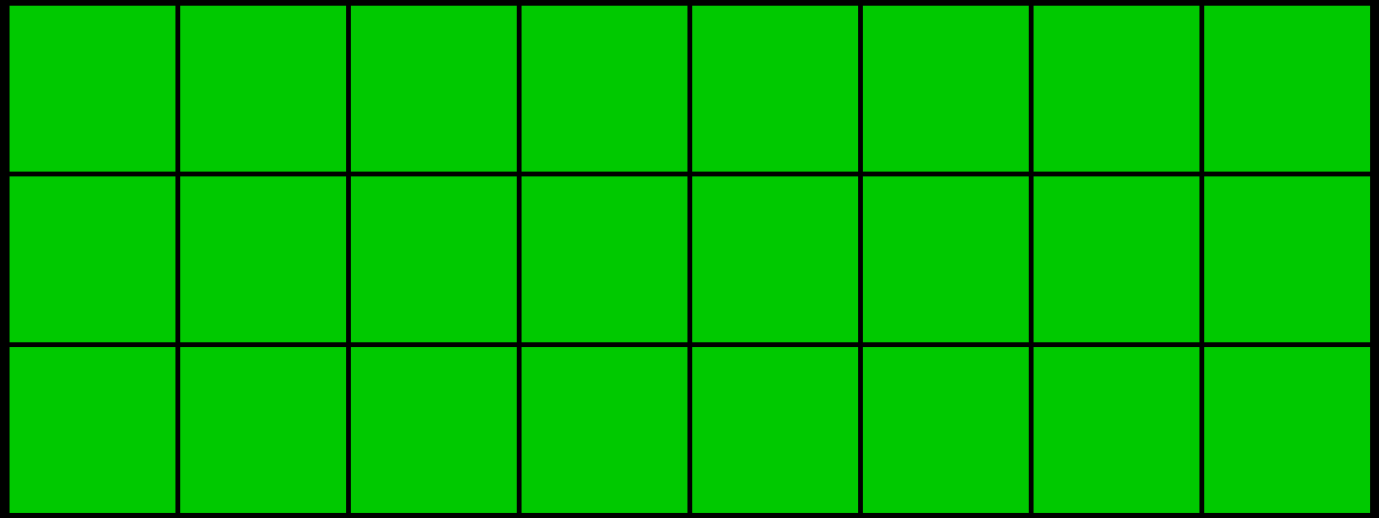


Memory deduplication

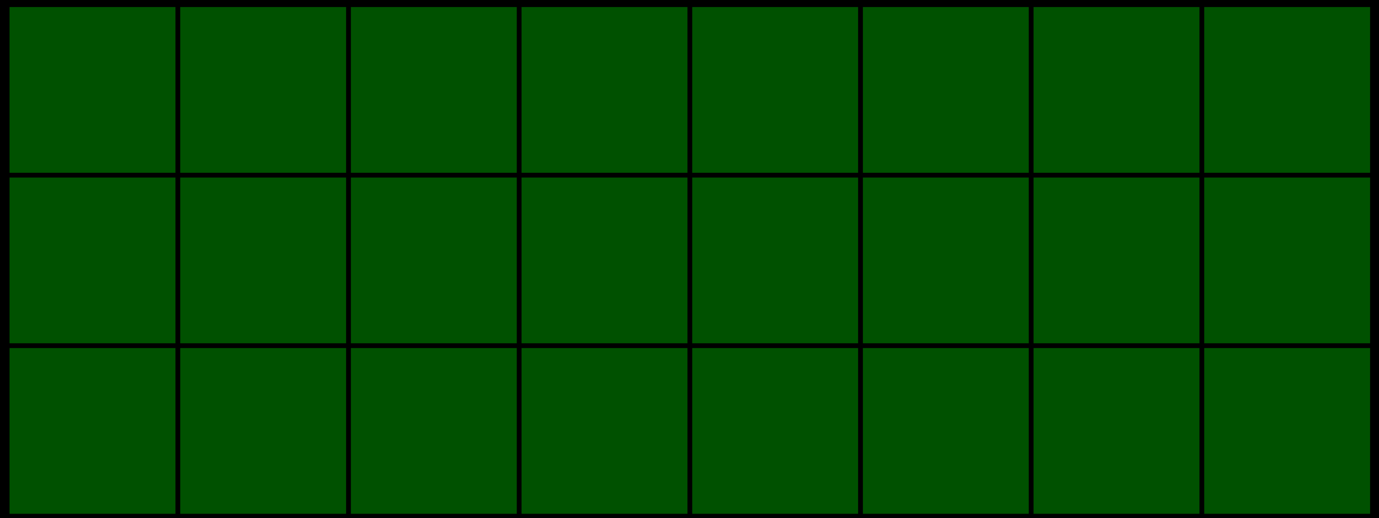
physical memory



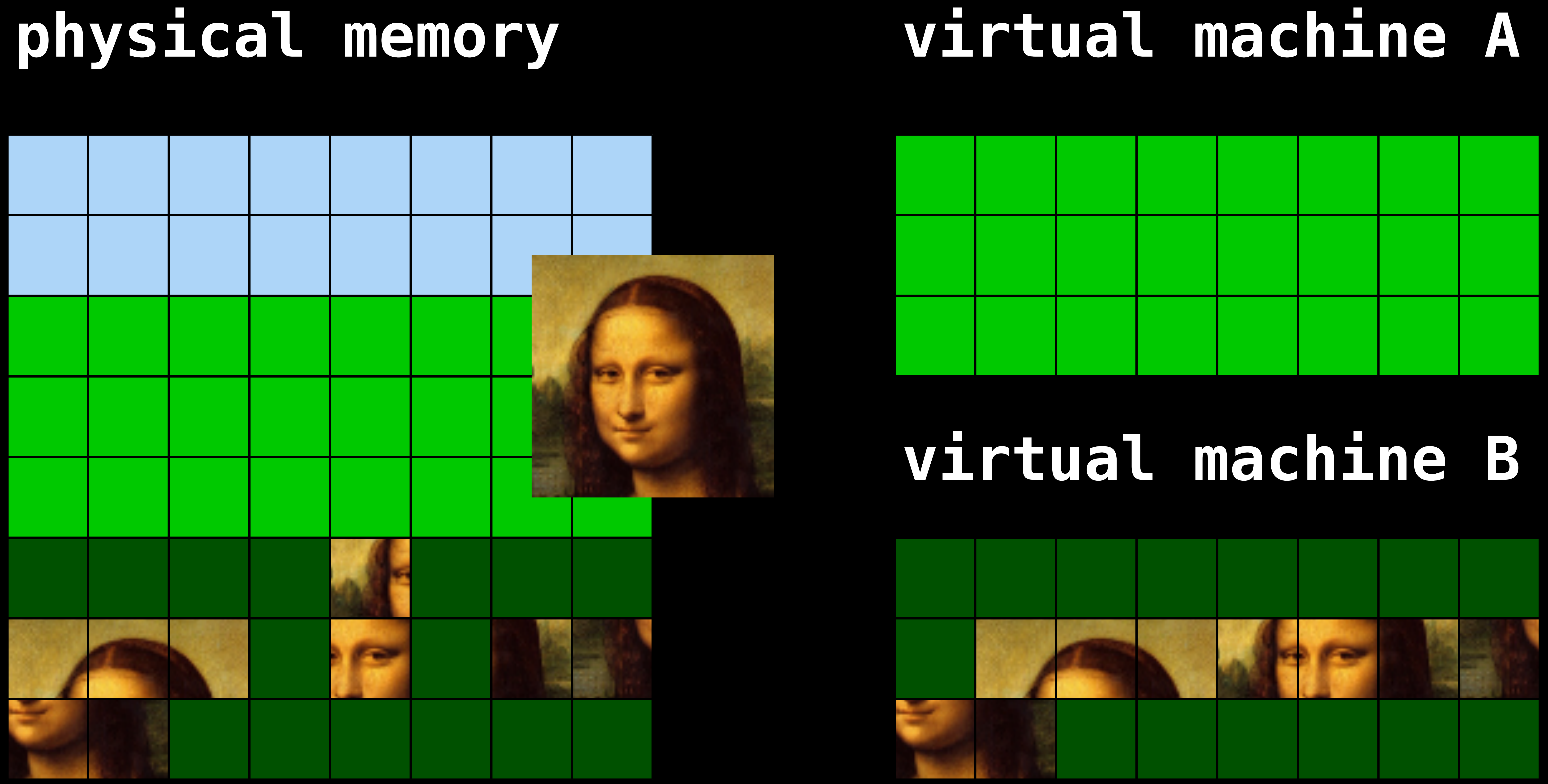
virtual machine A



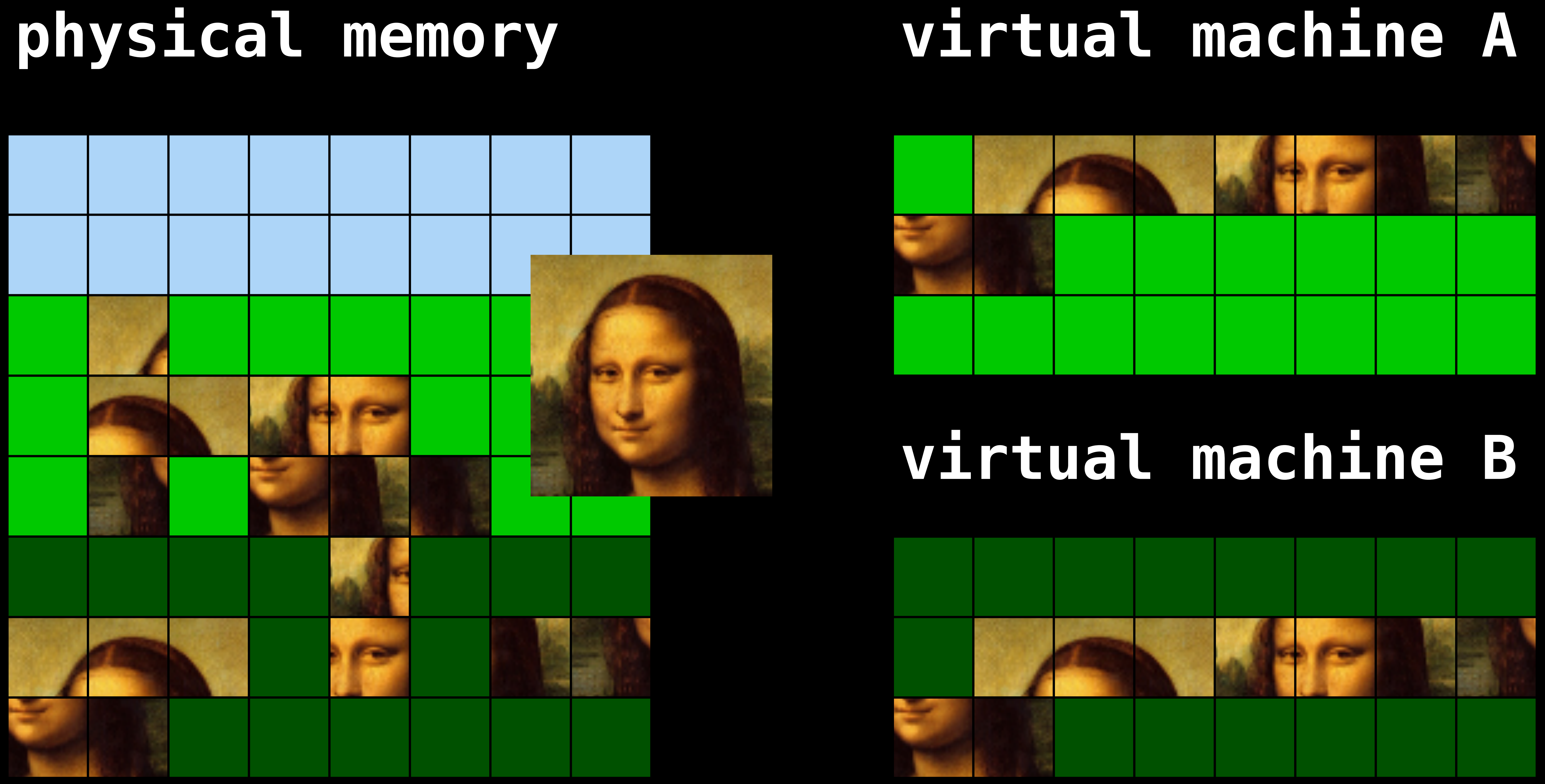
virtual machine B



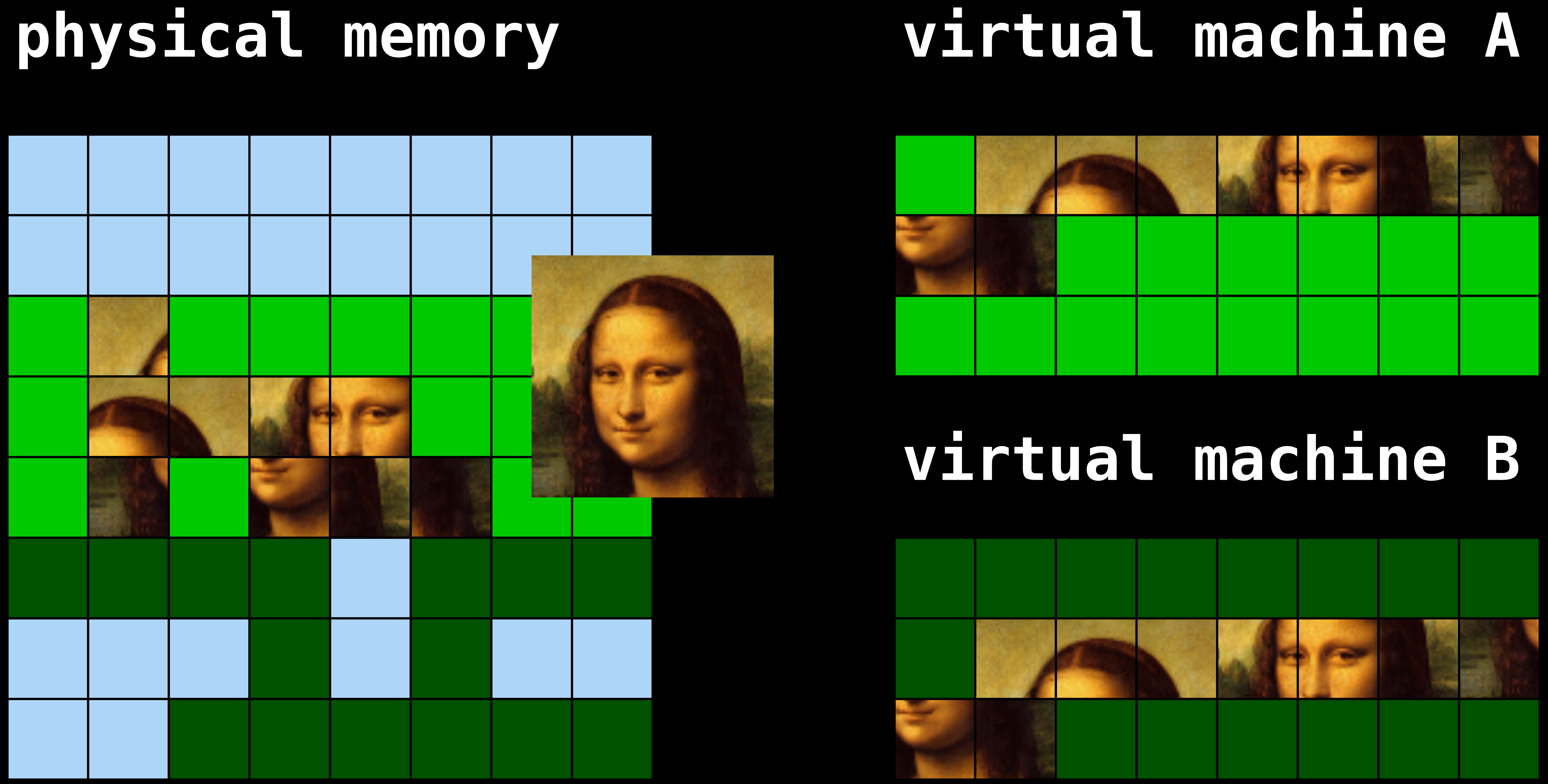
Memory deduplication



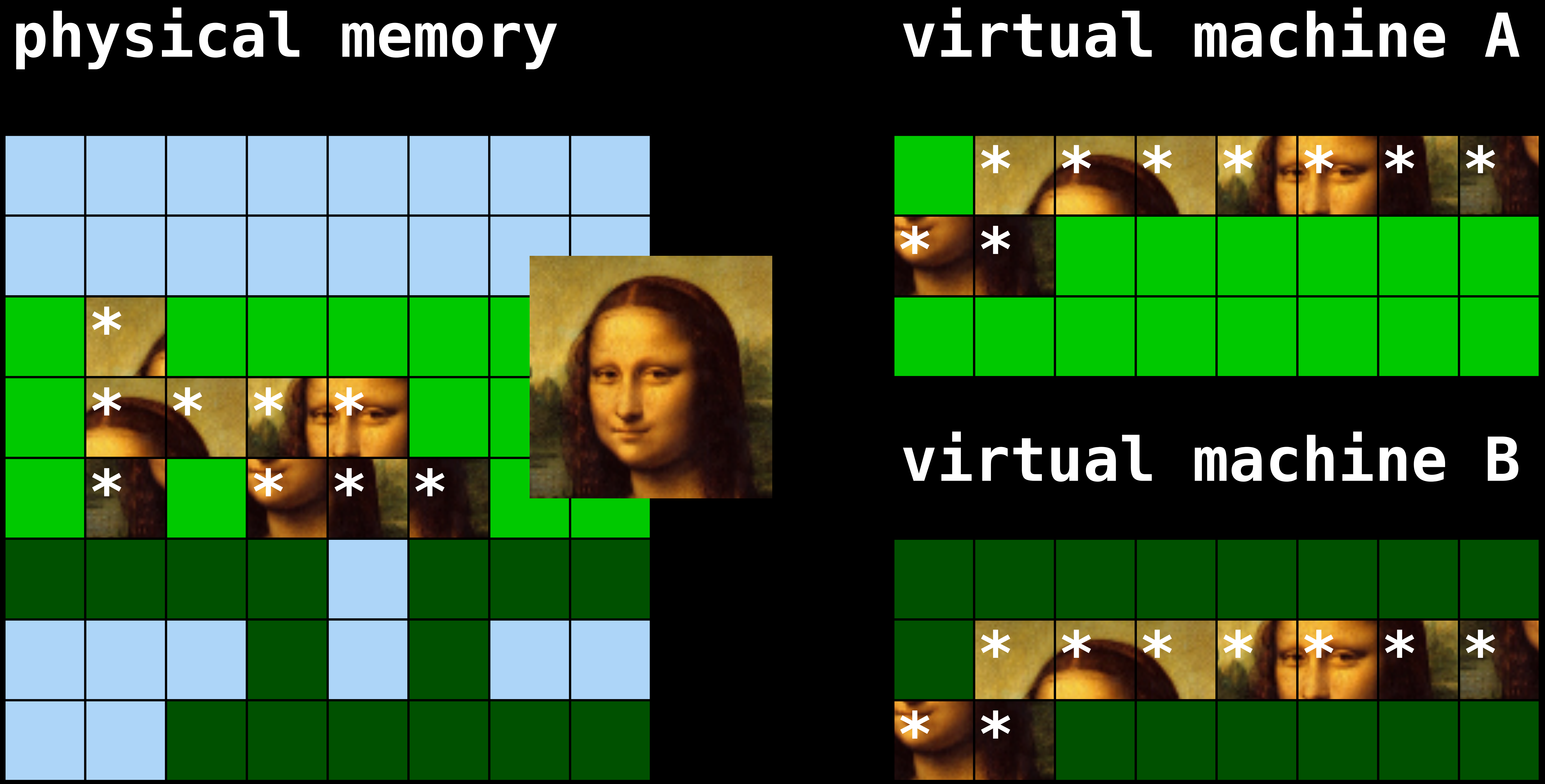
Memory deduplication



Memory deduplication



Memory deduplication



Kernel Same-page Merging (KSM)

- > Enabled by default for KVM (Ubuntu Server)
  - > Out-of-band Content Based Page Sharing (CBPS)

Kernel Same-page Merging (KSM)

- > Enabled by default for KVM (Ubuntu Server)
  - > Out-of-band Content Based Page Sharing (CBPS)

```
/sys/kernel/mm/ksm/run                   ‘1’ or ‘0’
/sys/kernel/mm/ksm/sleep_millisecs       e.g., 200 ms
/sys/kernel/mm/ksm/pages_to_scan         e.g., 100

1000/sleep_millisecs * pages_to_scan = pages per second
e.g., (1000/200ms) * 100 = 500 pages/sec
```

Memory deduplication: The Problem

Deduplicated memory does <u>not need</u> to have the <u>same security domain</u>.

(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

write

copy on write (due to deduplication)

trap

Deduplication side-channel attack

normal write

write

copy on write (due to deduplication)

trap

copy whole page

Deduplication side-channel attack

normal write

write

copy on write (due to deduplication)

trap

copy whole page

update page tables

Deduplication side-channel attack

normal write

write

copy on write (due to deduplication)

trap

copy whole page

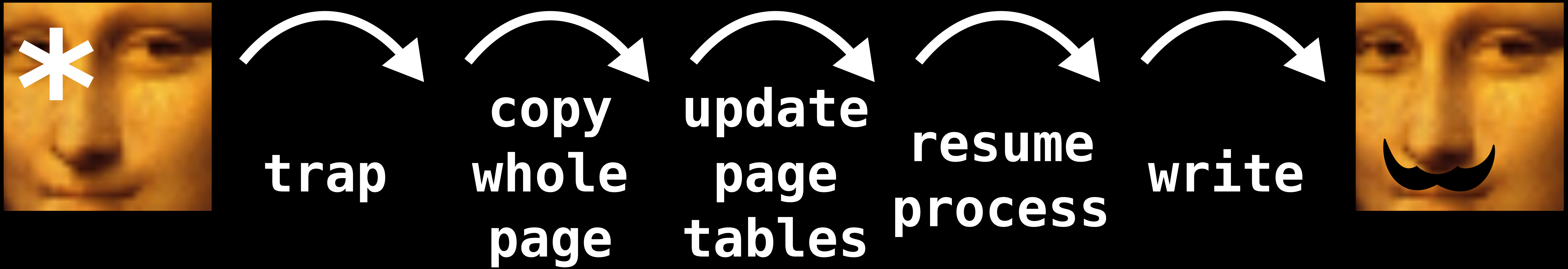
update page tables

resume process

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

Deduplication side-channel attack

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

> Cross-VM

Deduplication side-channel attack

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

> Cross-VM

> Cross-process

Deduplication side-channel attack

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

> Cross-VM

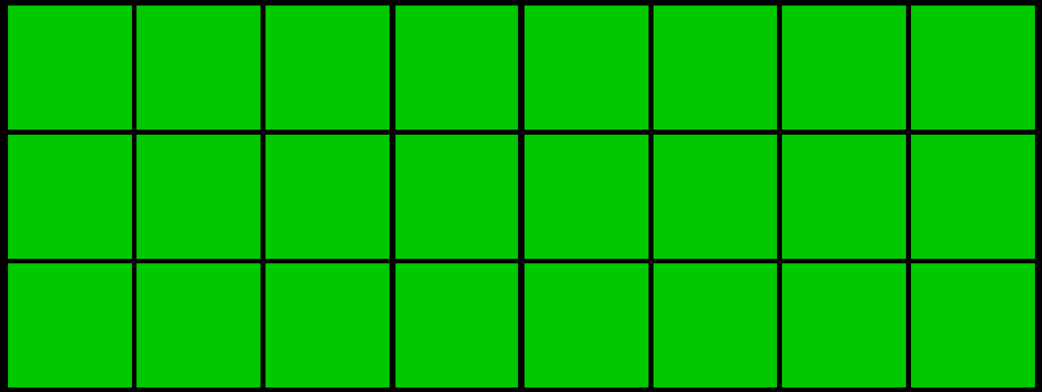
> Cross-process

> Intra-process, leak process data from JavaScript

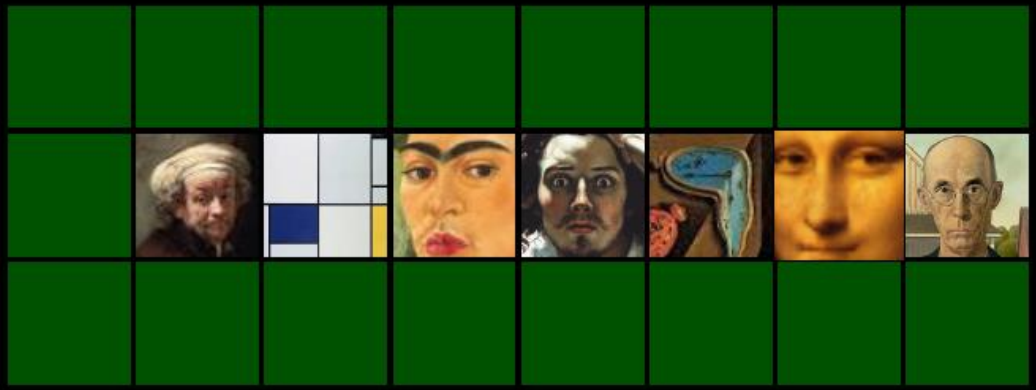
Exploitation of the side-channel

Exploitation of the side-channel

attacker memory



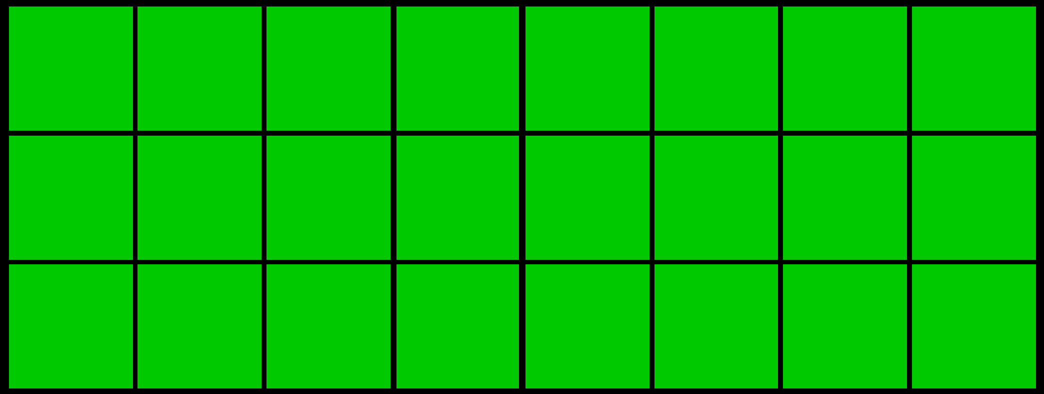
victim memory



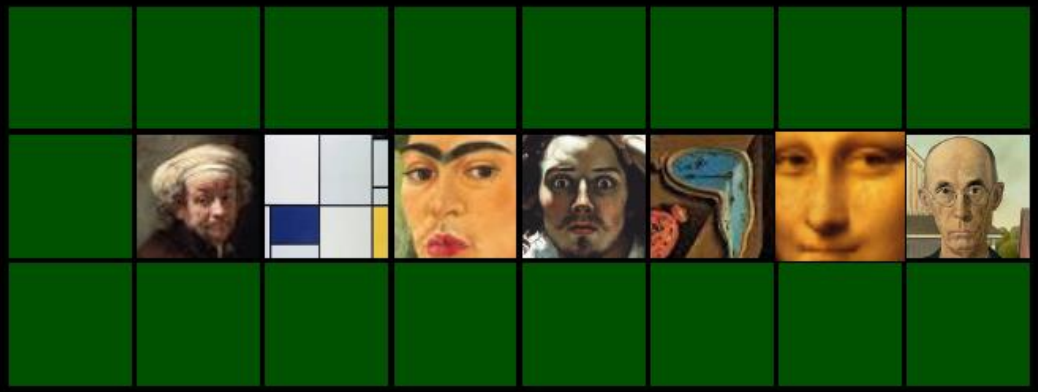
Exploitation of the side-channel

secret page

attacker memory

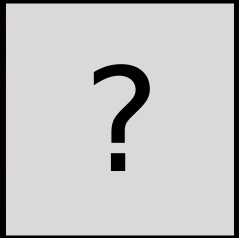


victim memory



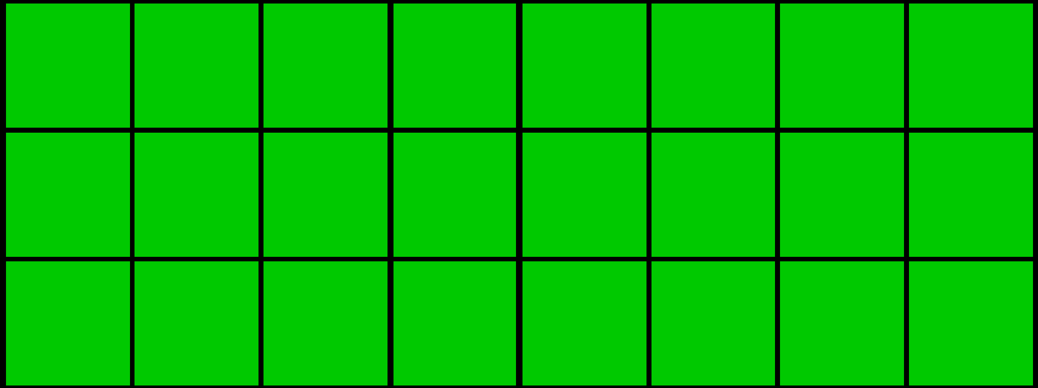
Exploitation of the side-channel

guess page

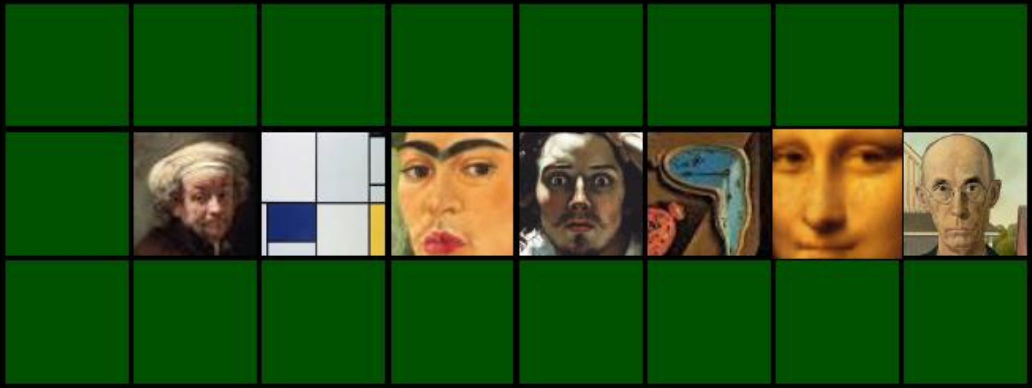


secret page

attacker memory



victim memory

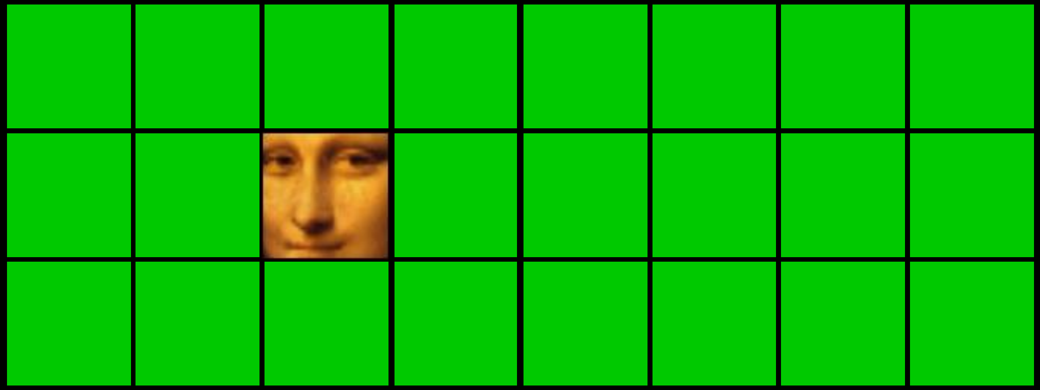


Exploitation of the side-channel

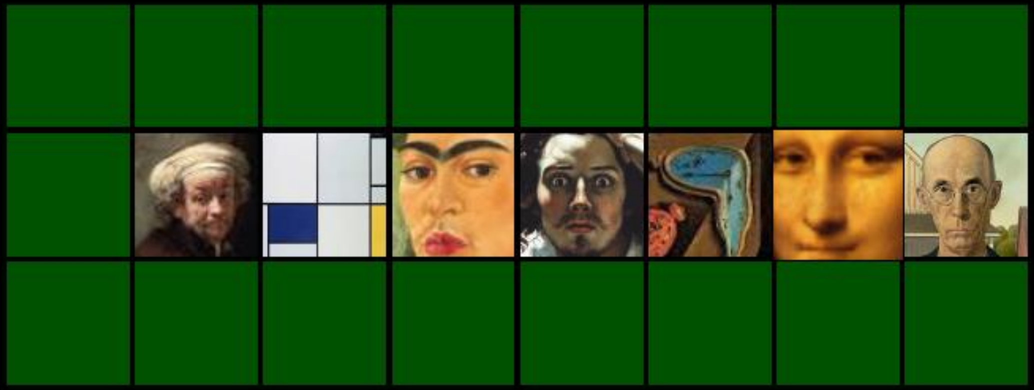
guess page

secret page

attacker memory



victim memory

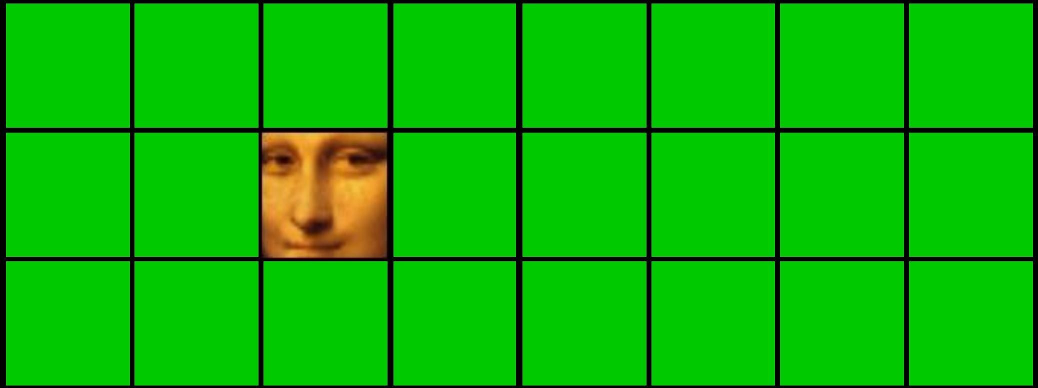


Exploitation of the side-channel

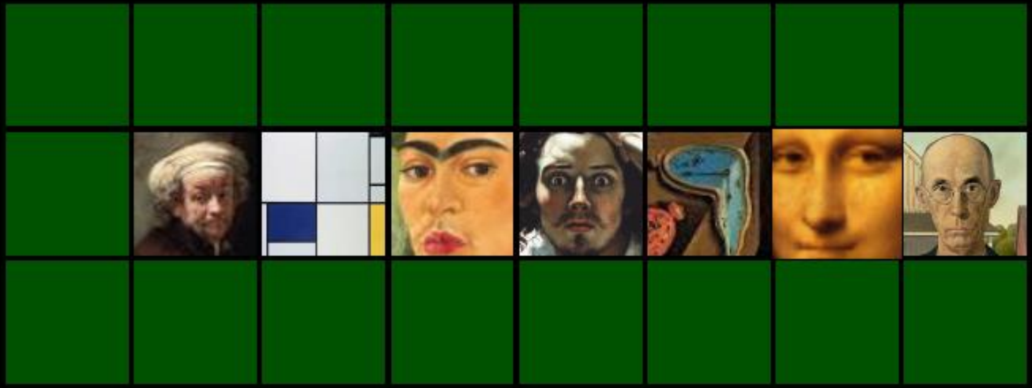
wait(t)

secret page

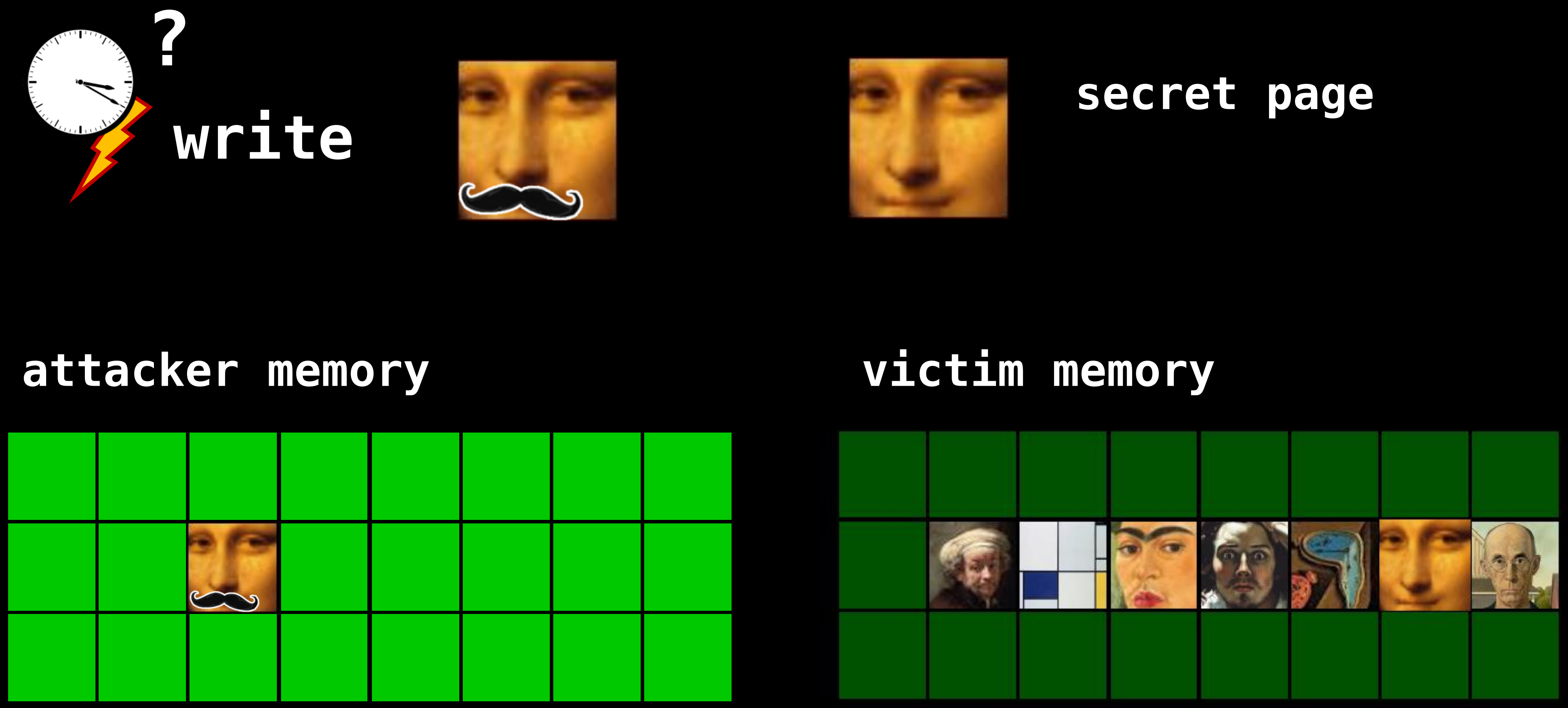
attacker memory



victim memory



Exploitation of the side-channel

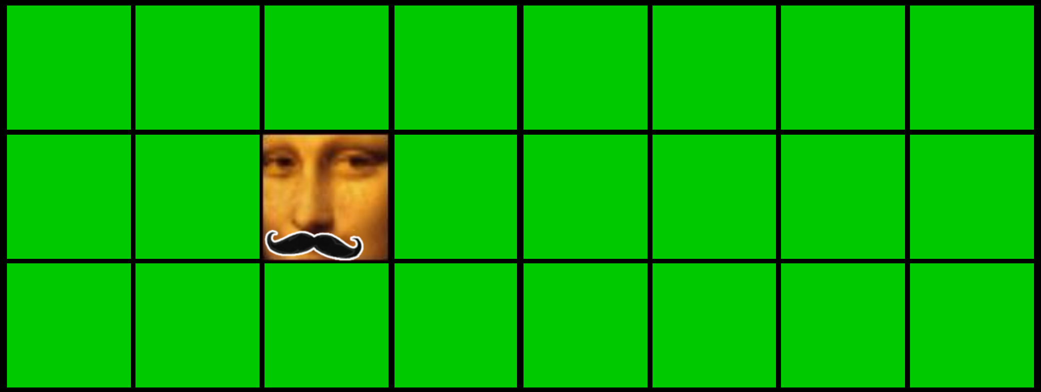


Exploitation of the side-channel

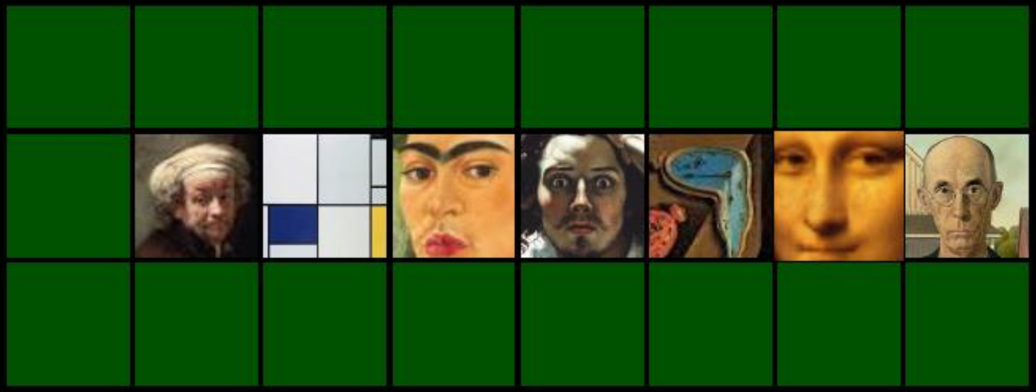
write time > threshold

secret page

attacker memory



victim memory

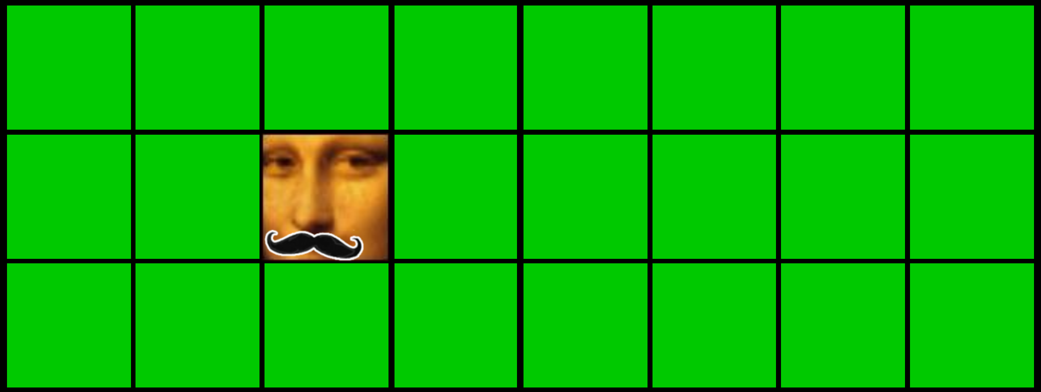


Exploitation of the side-channel

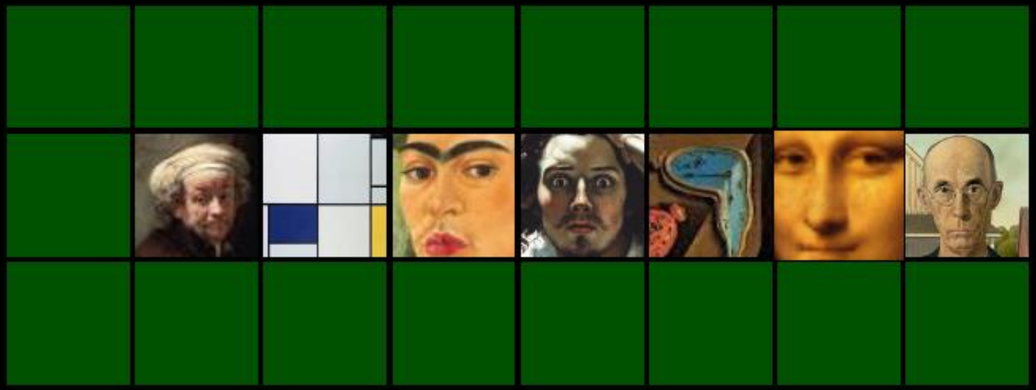
write time > threshold

secret page

attacker memory



victim memory

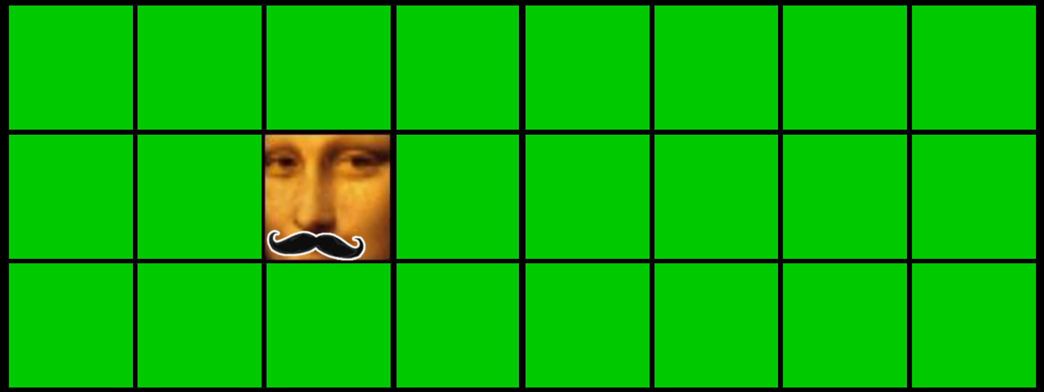


Exploitation of the side-channel

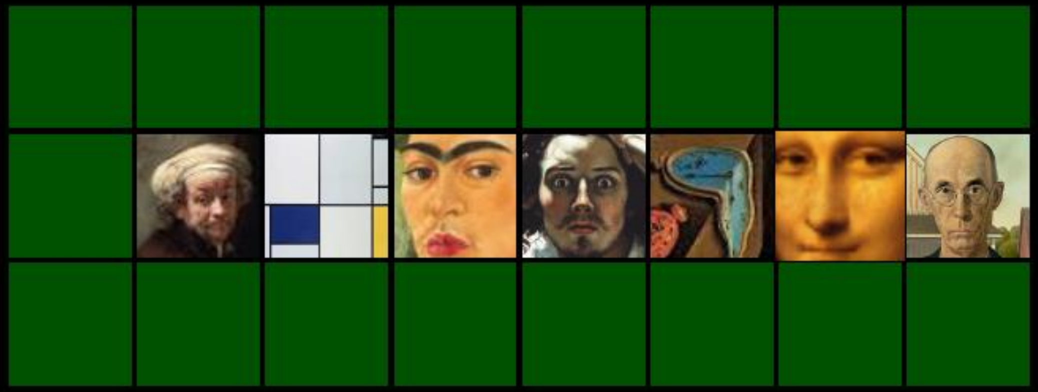
write time ≤ threshold

secret page

attacker memory



victim memory

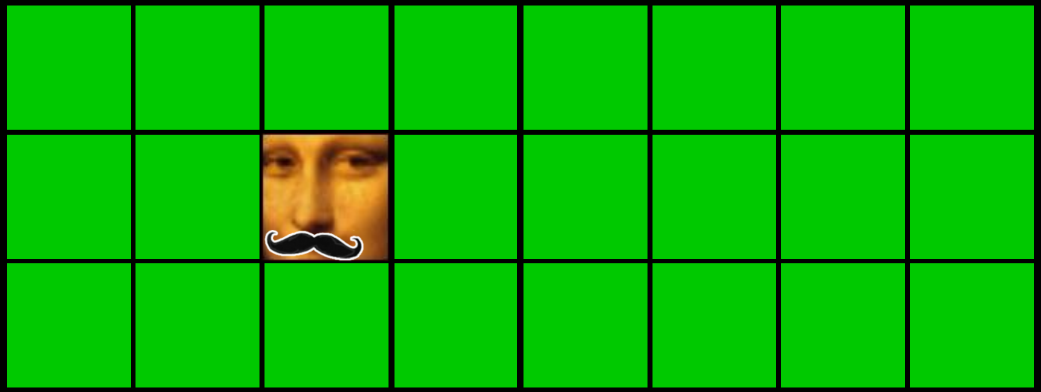


Exploitation of the side-channel

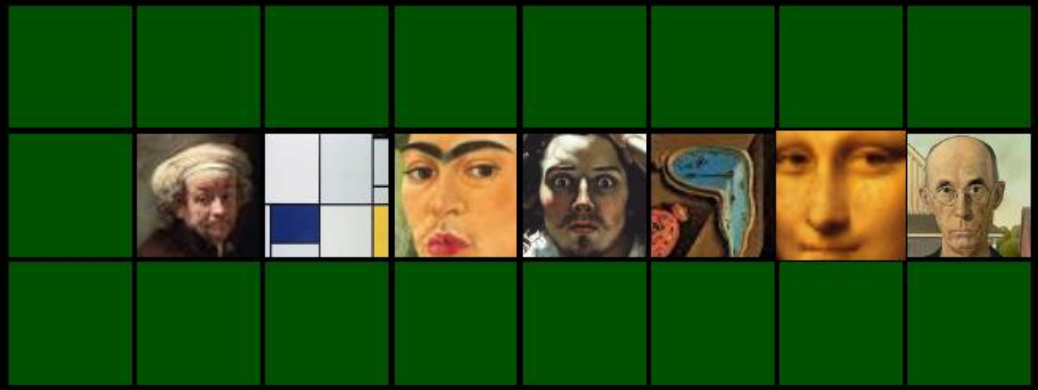
write time ≤ threshold

secret page

attacker memory



victim memory

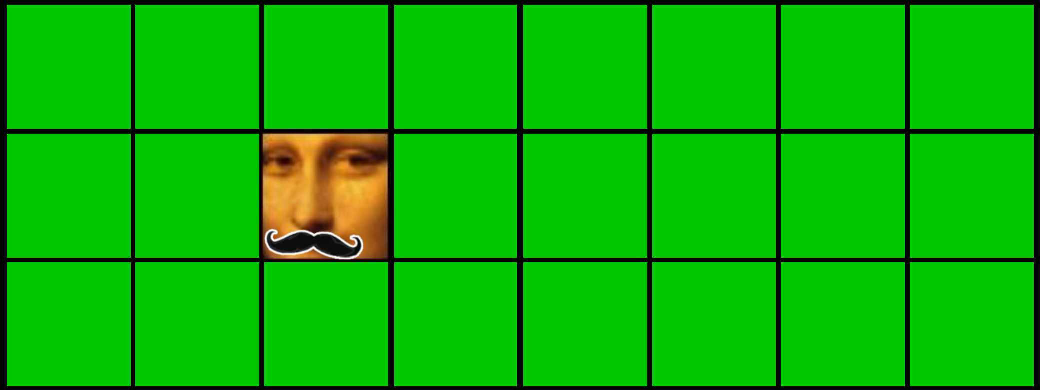


Exploitation of the side-channel

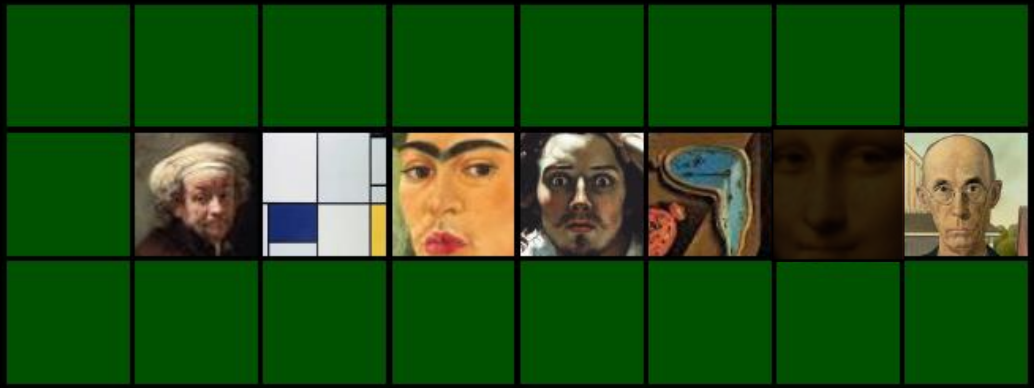
write time ≤ threshold

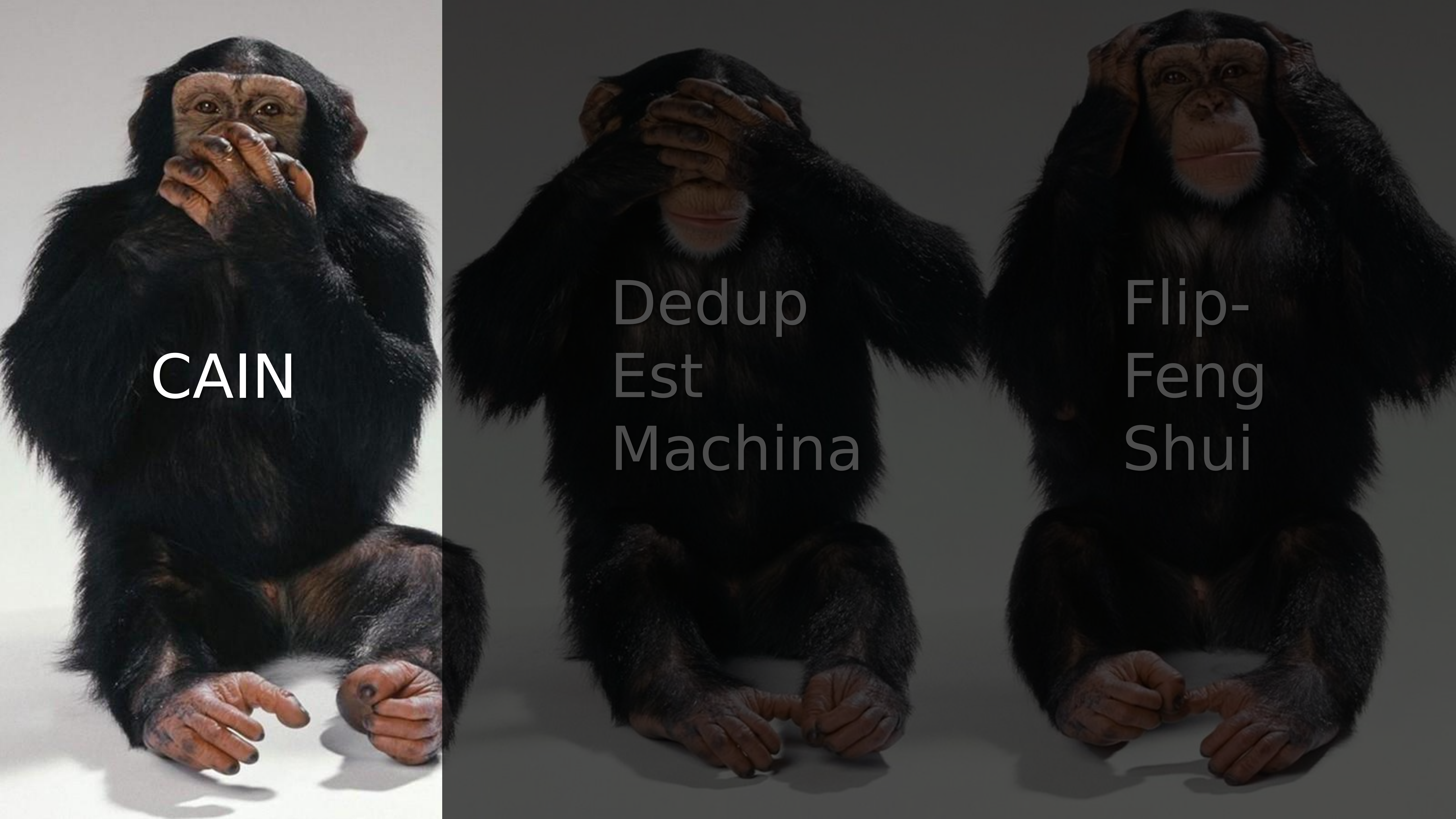
secret page

attacker memory



victim memory




CAIN
Dedup
Est
Machina
Flip-
Feng
Shui

CAIN:
Cross-VM Address Space Layout Introspection

Deduplication
(software side-channel)

CAIN:
Cross-VM Address Space Layout Introspection

Deduplication
(software side-channel)

↓

Cross-VM leak / ASLR bypass

CVE-2015-2877 / VU#935424 (https://www.kb.cert.org/vuls/id/935424)

CAIN

CAIN

> Page contents to leak ASLR? Secret page?

CAIN

> Page contents to leak ASLR? Secret page?

> How long to wait?

CAIN

> Page contents to leak ASLR? Secret page?

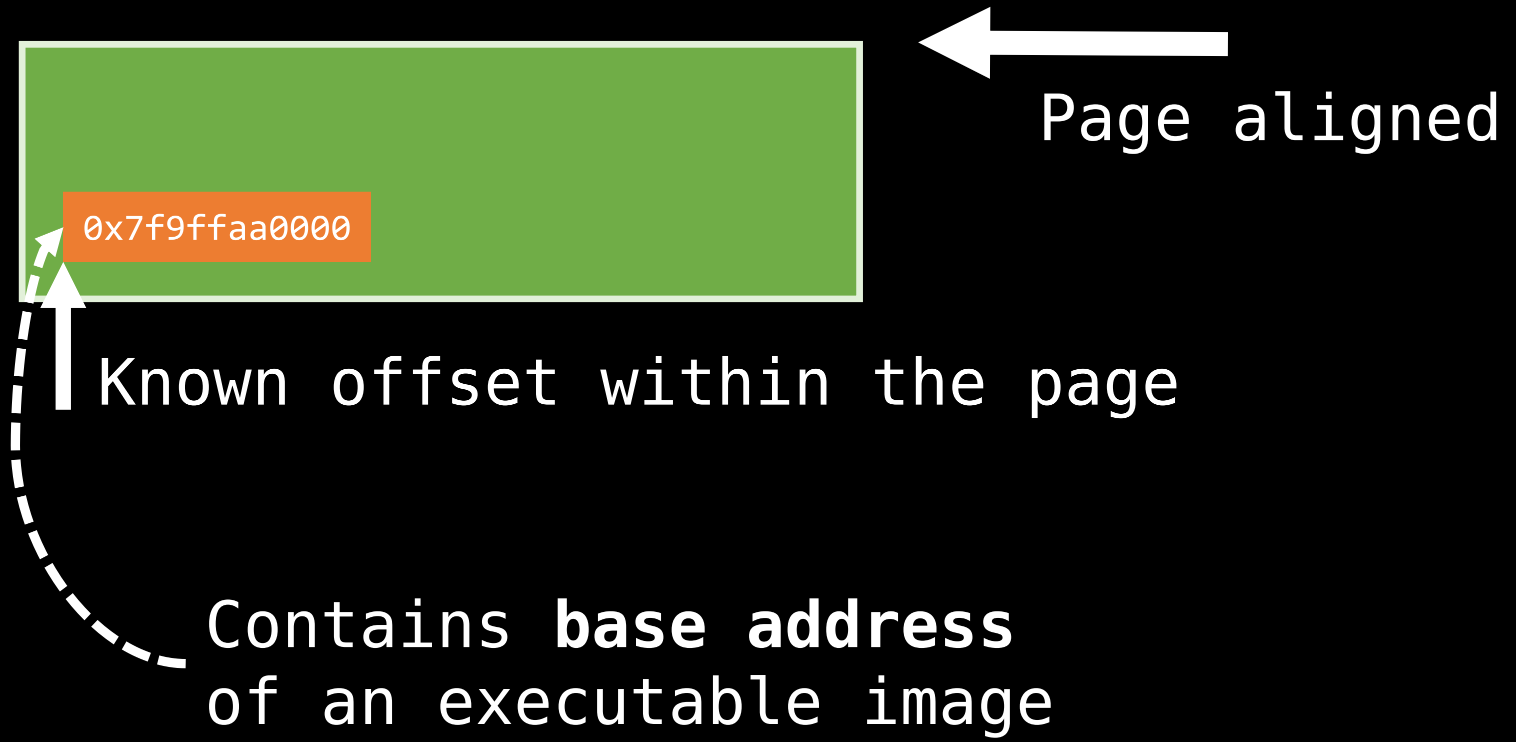
> How long to wait?

> How to detect a merged page? Noise?

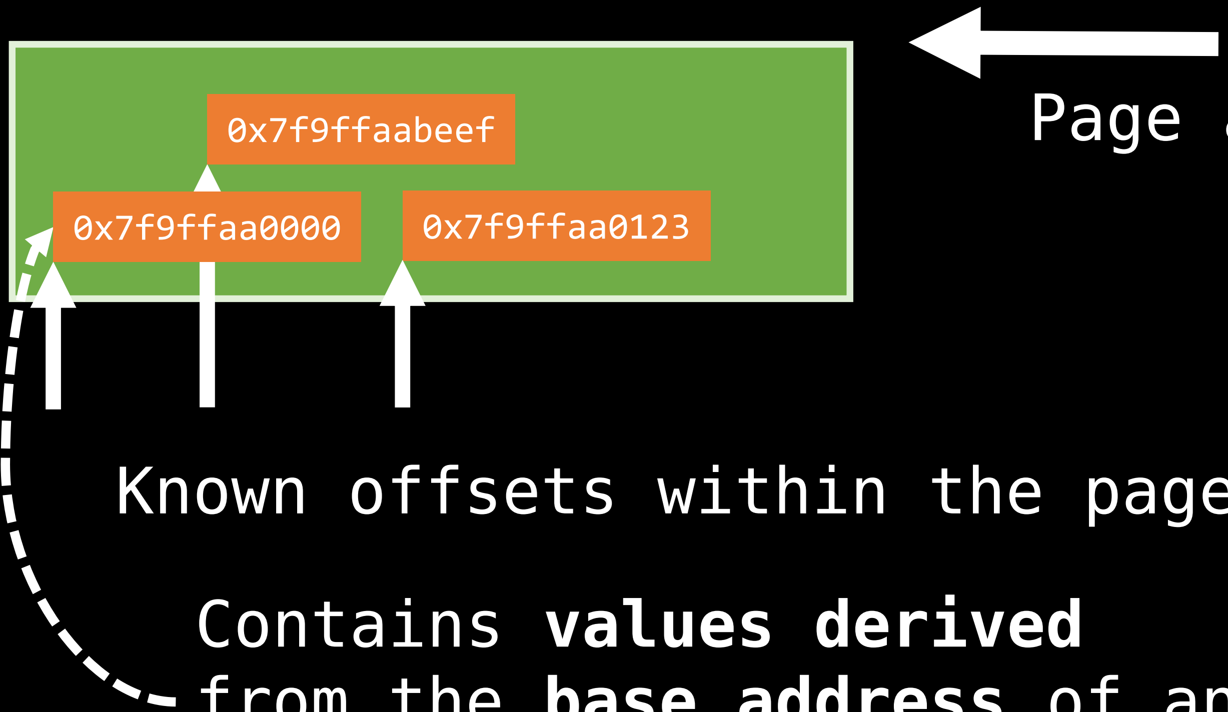
Suitable pages to break ASLR

- > Mostly static
- > Read-only in victim VM
- > Known to exist

Suitable pages to break ASLR



Suitable pages to break ASLR

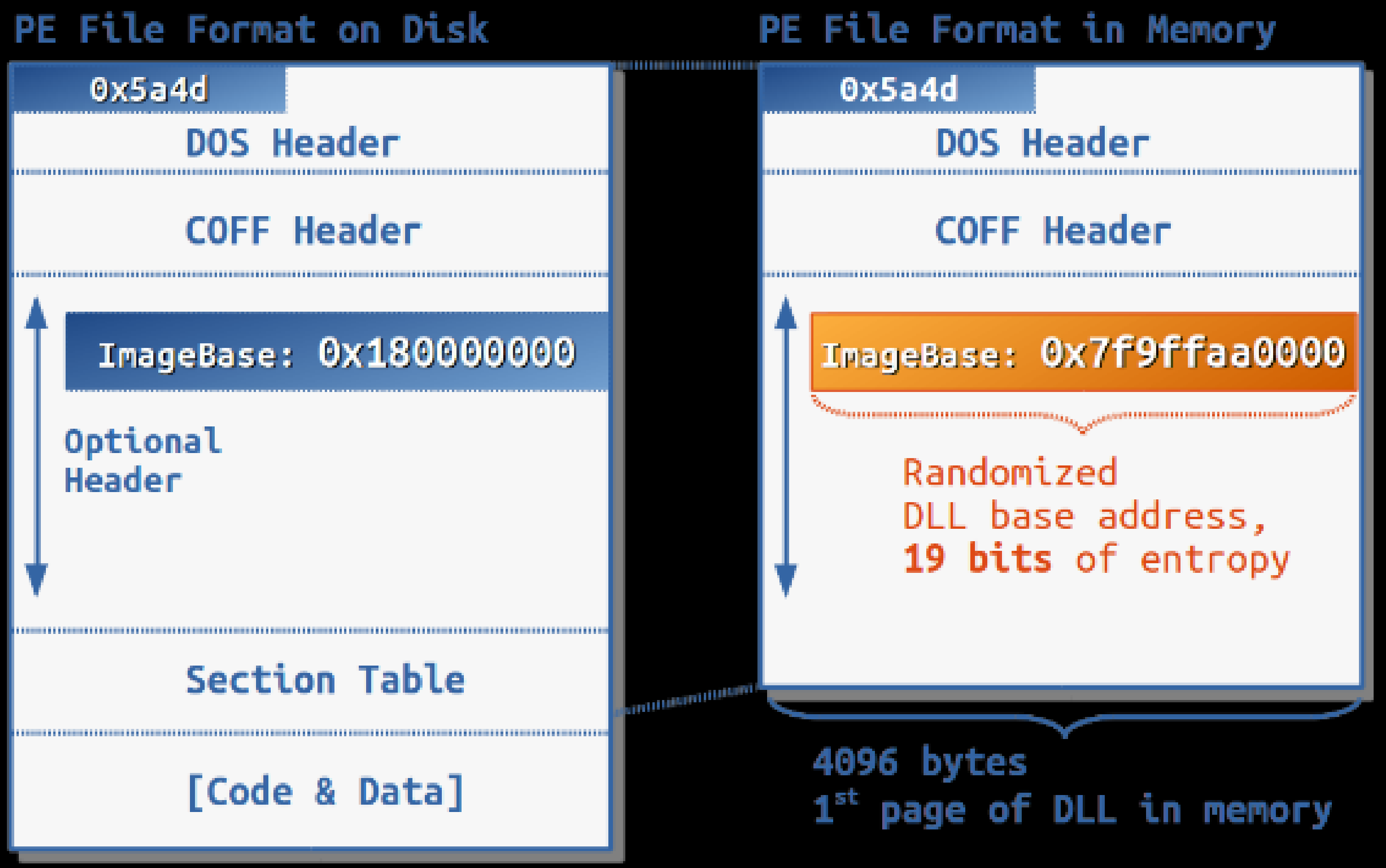


Page aligned

Known offsets within the page

Contains **values derived** from the **base address** of an executable image

Suitable page under Windows



Guessing the right address

> Well you still have to guess

Guessing the right address

> Well you still have to guess

    > 2^{19} base addresses for Windows x64

Guessing the right address

- > Well you still have to guess
    - > 2^{19} base addresses for Windows x64
    - > 524’288 guesses

Guessing the right address

- Well you still have to guess
  - 2^{19} base addresses for Windows x64
  - 524’288 guesses
  - One guess requires 1 page of memory

Based on http://sourceforge.net/projects/mpimd5bruteforc/

BRUTE FORCE

If it doesn't work, you're just not using enough.

Guessing the right address

> Attacker VM has much more memory

Guessing the right address

- > Attacker VM has much more memory
    - > Fill up memory with all guesses

Guessing the right address

- > Attacker VM has much more memory
  - > Fill up memory with all guesses
  - > 2^{19} * 1 page of 4 KB = 2 GB

Brute-force all addresses

<Page with RBA guess>
0x7f9ffa70000
0x7f9ffa80000
0x7f9ffa90000
0x7f9ffaa0000
0x7f9ffab0000
0x7f9ffac0000
0x7f9ffad0000

⋮

Brute-force all addresses

<Page with RBA guess>
0x7f9ffa70000
0x7f9ffa80000
0x7f9ffa90000
0x7f9ffaa0000
0x7f9ffab0000
0x7f9ffac0000
0x7f9ffad0000
⋮

detect_shared_pages()

0x7f9ffaa0000

Wait for how long?

Wait for how long?

> Depends on the memory deduplication implementation

Wait for how long?

> Depends on the memory deduplication implementation

> Varies depending on amount of memory used

Wait for how long?

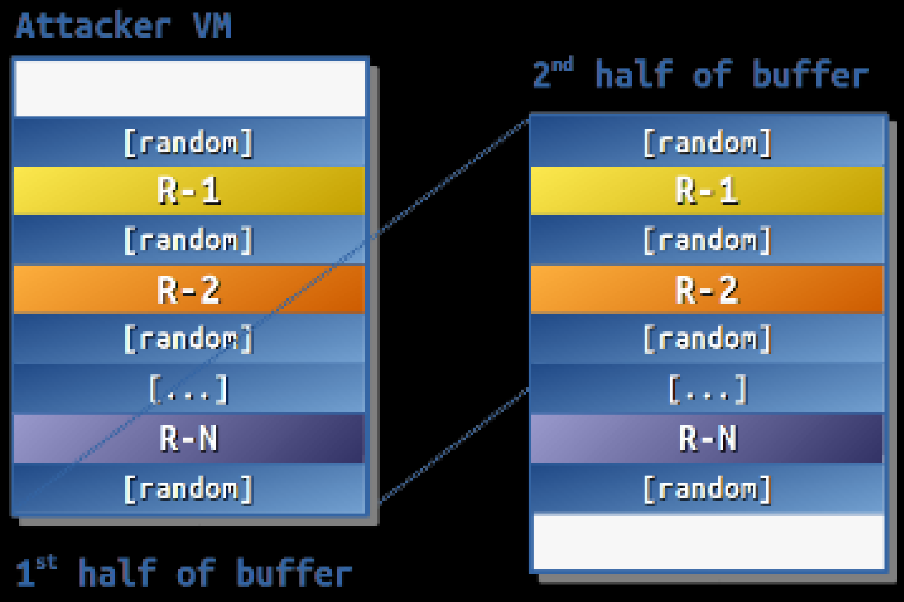
- Depends on the memory deduplication implementation
- Varies depending on amount of memory used
- Attacker trade-off
  - Waiting too little obstructs the attack
  - Waiting too long increases attack time

Adaptive sleep-time detection

> Try to automatically detect sleep time

Adaptive sleep-time detection

> Try to automatically detect sleep time



Adaptive sleep-time detection

> Try to automatically detect sleep time

> After buffer creation, wait e.g. t = 10min

Adaptive sleep-time detection

> Try to automatically detect sleep time

> After buffer creation, wait e.g. t = 10min

    > Detect how many pages were merged

Adaptive sleep-time detection

> Try to automatically detect sleep time

> After buffer creation, wait e.g. t = 10min

> Detect how many pages were merged

> If detection rate > threshold (e.g. 90%)

Adaptive sleep-time detection

> Try to automatically detect sleep time

> After buffer creation, wait e.g. t = 10min

- > Detect how many pages were merged
- > If detection rate > threshold (e.g. 90%)
  - > Use t

Adaptive sleep-time detection

> Try to automatically detect sleep time

> After buffer creation, wait e.g. t = 10min

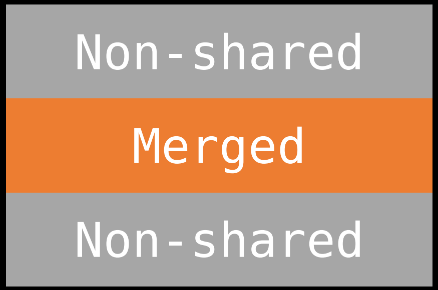
    > Detect how many pages were merged

    > If detection rate > threshold (e.g. 90%)

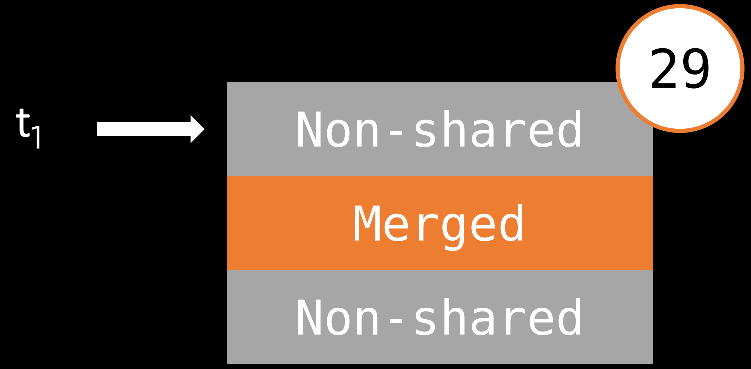
        > Use t

    > Else, increase t and try again

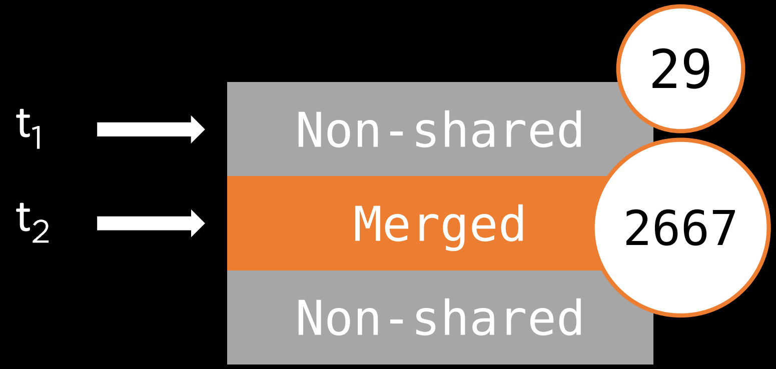
Detect merged pages



Detect merged pages

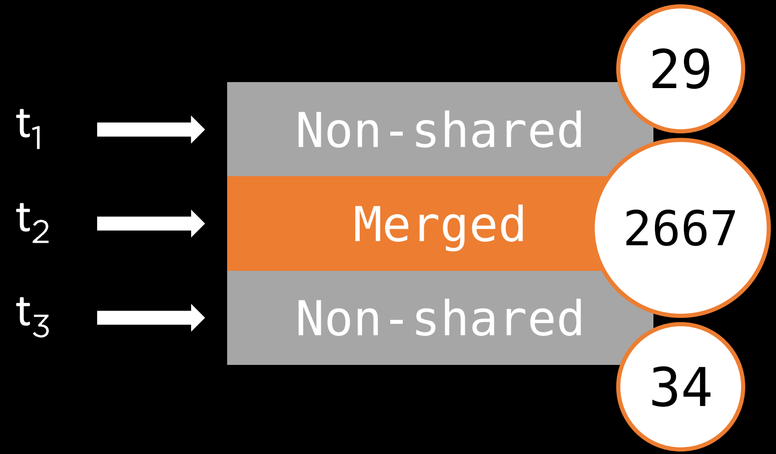


Detect merged pages



Detect merged pages

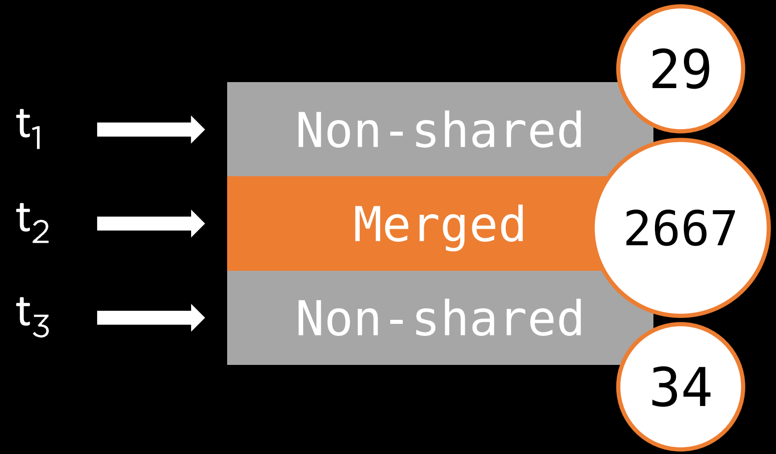
Measure write time with rdtsc
(Read Time Stamp Counter)



Detect merged pages

Measure write time with rdtsc

(Read Time Stamp Counter)



$t_2 > 2 * (t_1+t_3)/2$ $t_{1,3} < M = 1000$ $t_1 < t_3, (t_3-t_1) < t_3/3$

Detect merged pages

These heuristics worked for different HW configurations

$t_2 > 2$ … 1000 $t_1 < t_3$, $(t_3 - t_1) < t_3/3$

Handling noise

> Be conservative and perform multiple rounds

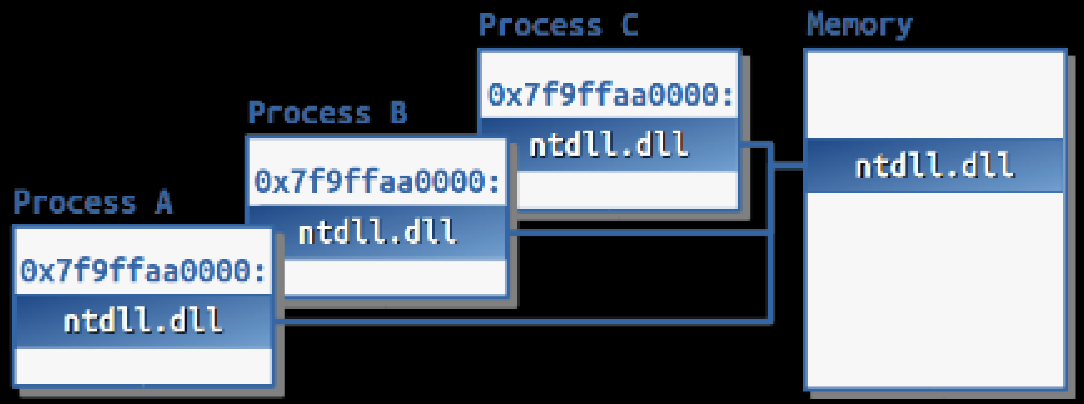
Handling noise

> Be conservative and perform multiple rounds

  > Probability that same guess is affected by noise in different rounds is low

Windows x64 ASLR

- High Entropy ASLR
  - 33 bits for stacks
  - 24 bits for heaps

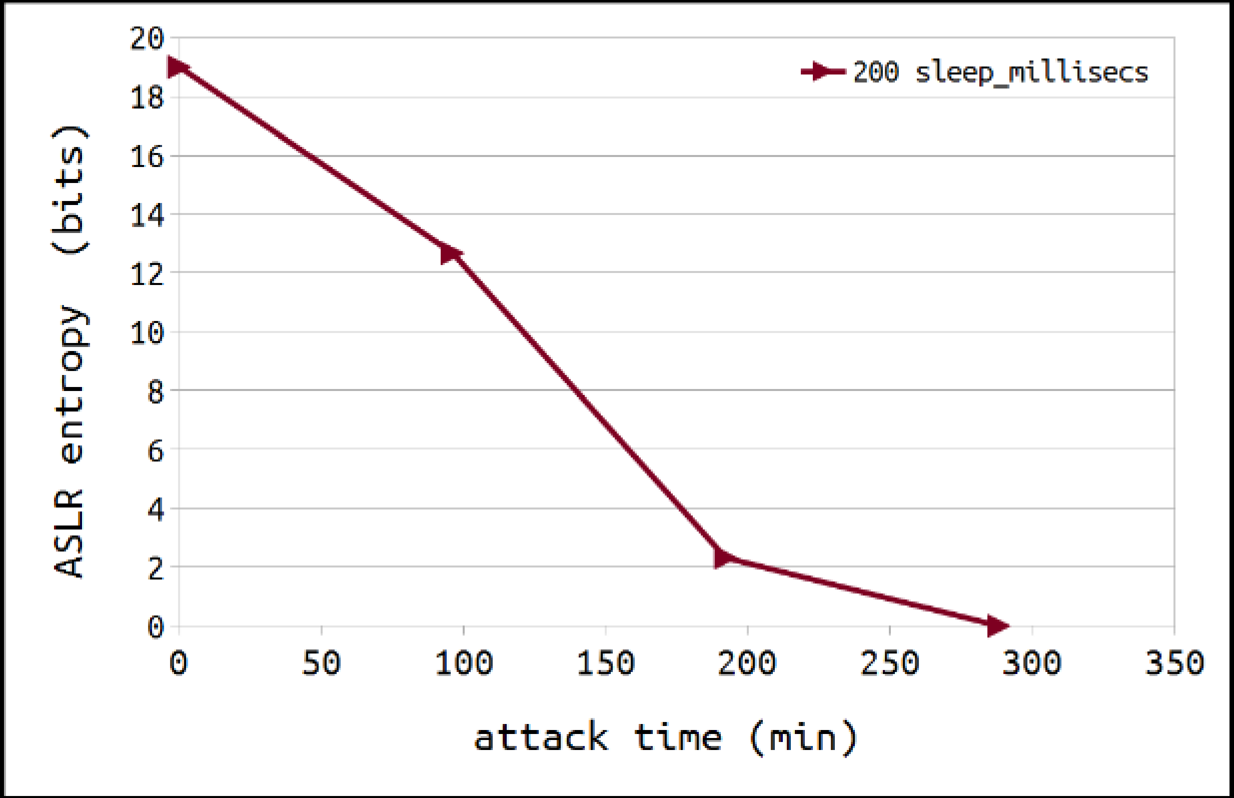


- 17 bits for executables
- 19 bits for DLLS

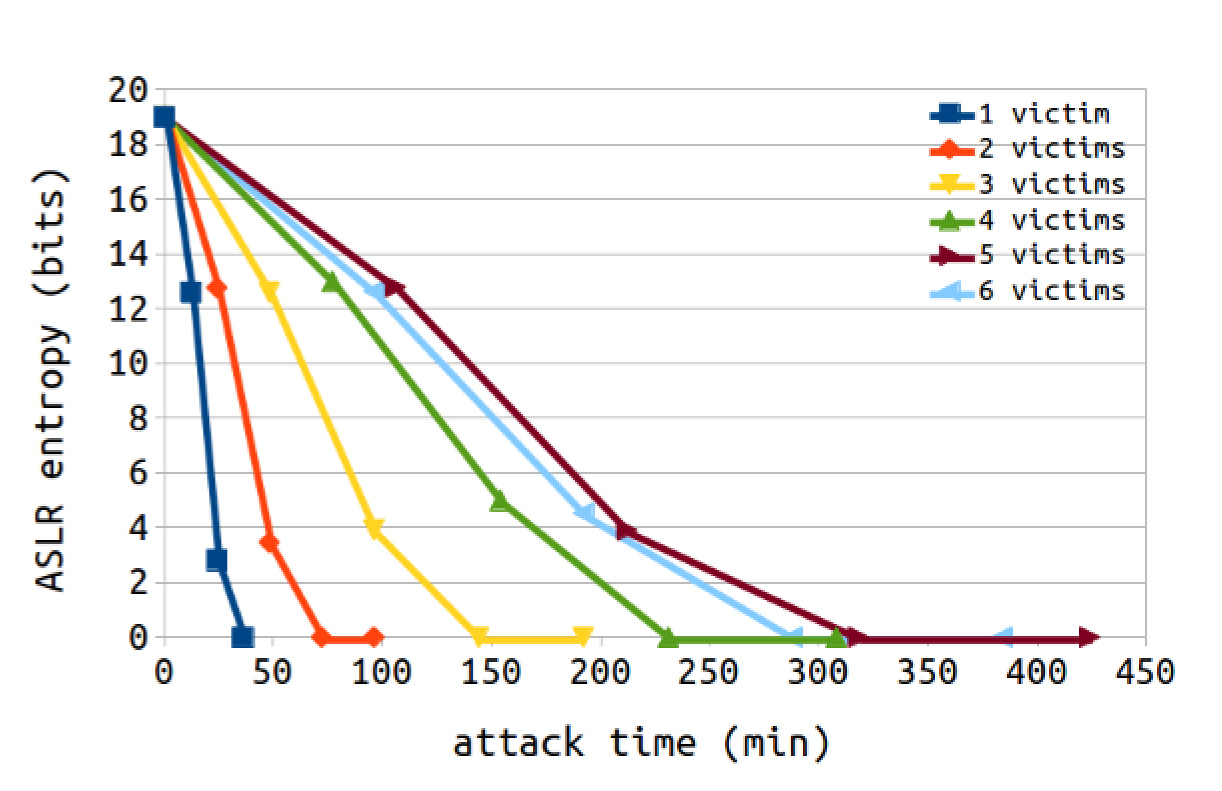
System-wide at boot-time for certain images

http://www.nynaeve.net/?p=198
https://media.blackhat.com/bh-us-12/Briefings/M_Miller/BH_US_12_Miller_Exploit_Mitigation_Slides.pdf

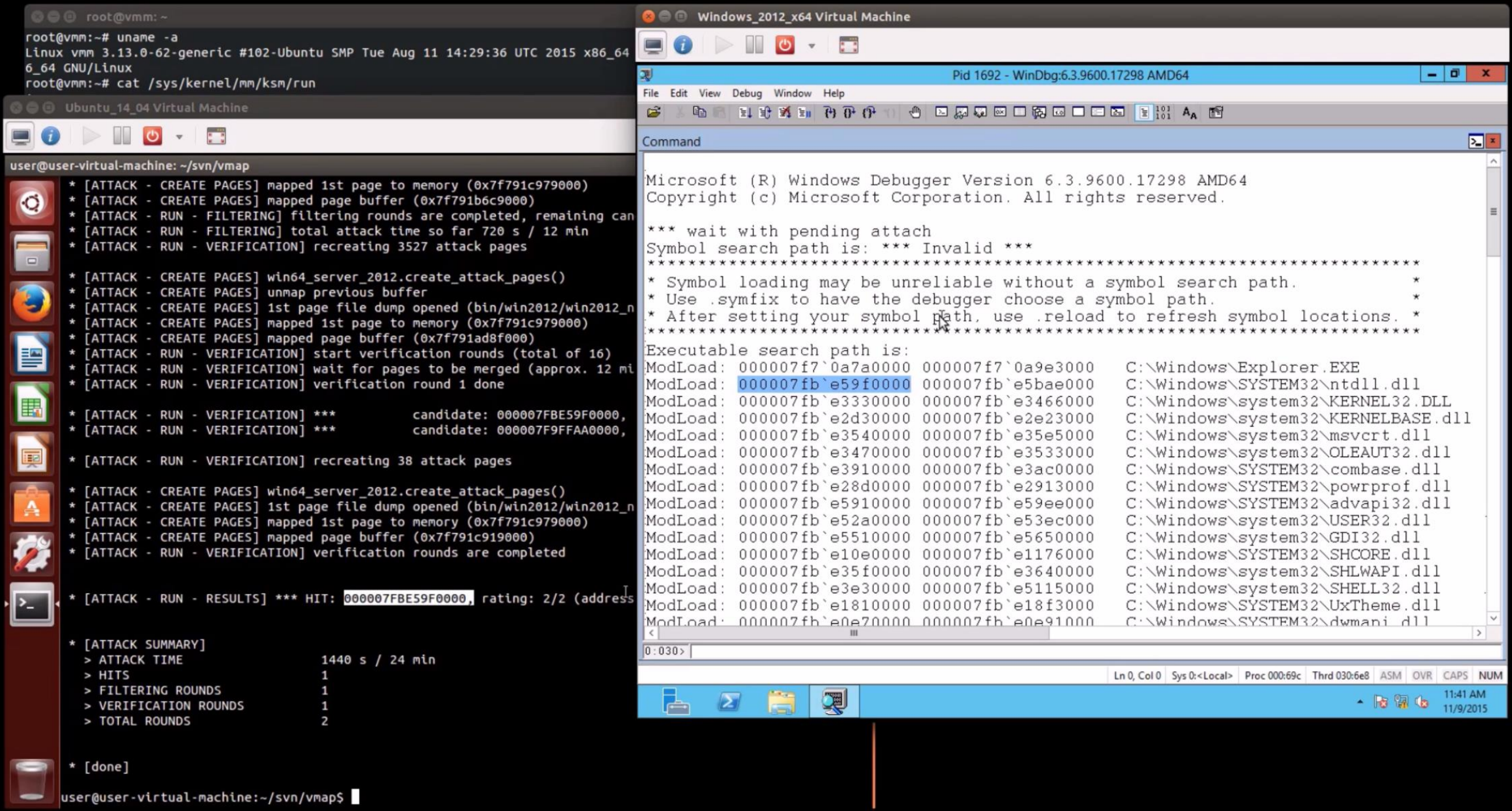
Attacking a single Windows VM

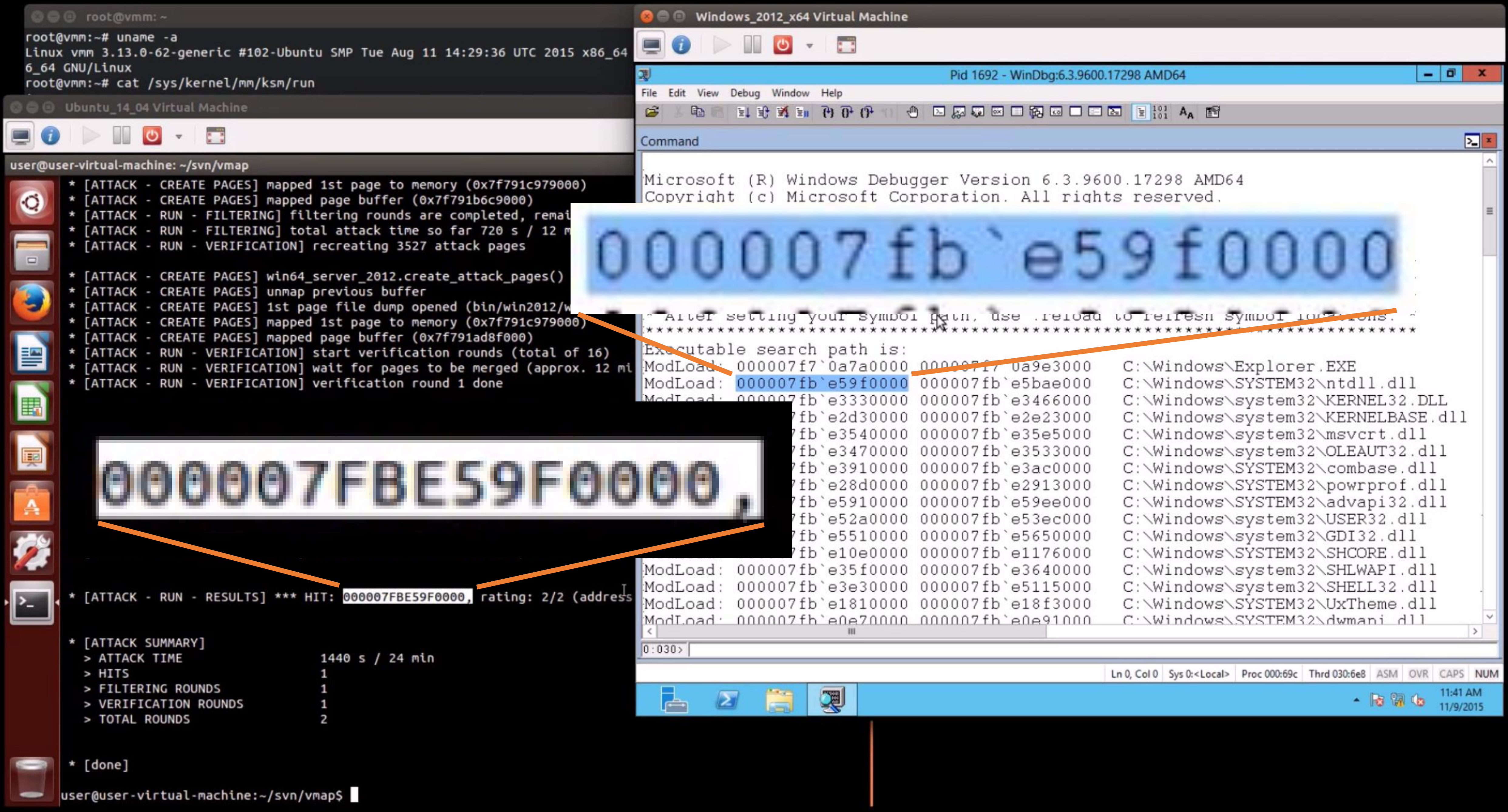


Attacking multiple Windows VM



sleep_millisecs = 20


root@vmm: ~
root@vmm:~# uname -a
Linux vmm 3.13.0-62-generic #102-Ubuntu SMP Tue Aug 11 14:29:36 UTC 2015 x86_64
6_64 GNU/Linux
root@vmm:~# cat /sys/kernel/mm/ksm/run
Ubuntu_14_04 Virtual Machine
user@user-virtual-machine: ~/svn/vmap
* [ATTACK - CREATE PAGES] mapped 1st page to memory (0x7f791c979000)
* [ATTACK - CREATE PAGES] mapped page buffer (0x7f791b6c9000)
* [ATTACK - RUN - FILTERING] filtering rounds are completed, remaining can
* [ATTACK - RUN - FILTERING] total attack time so far 720 s / 12 min
* [ATTACK - RUN - VERIFICATION] recreating 3527 attack pages
* [ATTACK - CREATE PAGES] win64_server_2012.create_attack_pages()
* [ATTACK - CREATE PAGES] unmap previous buffer
* [ATTACK - CREATE PAGES] 1st page file dump opened (bin/win2012/win2012_n
* [ATTACK - CREATE PAGES] mapped 1st page to memory (0x7f791c979000)
* [ATTACK - CREATE PAGES] mapped page buffer (0x7f791ad8f000)
* [ATTACK - RUN - VERIFICATION] start verification rounds (total of 16)
* [ATTACK - RUN - VERIFICATION] wait for pages to be merged (approx. 12 mi
* [ATTACK - RUN - VERIFICATION] verification round 1 done
* [ATTACK - RUN - VERIFICATION] *** candidate: 000007FBE59F0000,
* [ATTACK - RUN - VERIFICATION] *** candidate: 000007F9FFAA0000,
* [ATTACK - RUN - VERIFICATION] recreating 38 attack pages
* [ATTACK - CREATE PAGES] win64_server_2012.create_attack_pages()
* [ATTACK - CREATE PAGES] 1st page file dump opened (bin/win2012/win2012_n
* [ATTACK - CREATE PAGES] mapped 1st page to memory (0x7f791c979000)
* [ATTACK - CREATE PAGES] mapped page buffer (0x7f791c919000)
* [ATTACK - RUN - VERIFICATION] verification rounds are completed
* [ATTACK - RUN - RESULTS] *** HIT: 000007FBE59F0000, rating: 2/2 (address
* [ATTACK SUMMARY]
> ATTACK TIME 1440 s / 24 min
> HITS 1
> FILTERING ROUNDS 1
> VERIFICATION ROUNDS 1
> TOTAL ROUNDS 2
* [done]
user@user-virtual-machine:~/svn/vmap$
Windows_2012_x64 Virtual Machine
Pid 1692 - WinDbg:6.3.9600.17298 AMD64
File Edit View Debug Window Help
Command
Microsoft (R) Windows Debugger Version 6.3.9600.17298 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.
*** wait with pending attach
Symbol search path is: *** Invalid ***
* Symbol loading may be unreliable without a symbol search path. *
* Use .symfix to have the debugger choose a symbol path. *
* After setting your symbol path, use .reload to refresh symbol locations. *
Executable search path is:
ModLoad: 000007f7`0a7a0000 000007f7`0a9e3000 C:\Windows\Explorer.EXE
ModLoad: 000007fb`e59f0000 000007fb`e5bae000 C:\Windows\SYSTEM32\ntdll.dll
ModLoad: 000007fb`e3330000 000007fb`e3466000 C:\Windows\system32\KERNEL32.DLL
ModLoad: 000007fb`e2d30000 000007fb`e2e23000 C:\Windows\system32\KERNELBASE.dll
ModLoad: 000007fb`e3540000 000007fb`e35e5000 C:\Windows\system32\msvcrt.dll
ModLoad: 000007fb`e3470000 000007fb`e3533000 C:\Windows\system32\OLEAUT32.dll
ModLoad: 000007fb`e3910000 000007fb`e3ac0000 C:\Windows\SYSTEM32\combase.dll
ModLoad: 000007fb`e28d0000 000007fb`e2913000 C:\Windows\SYSTEM32\powrprof.dll
ModLoad: 000007fb`e5910000 000007fb`e59ee000 C:\Windows\SYSTEM32\advapi32.dll
ModLoad: 000007fb`e52a0000 000007fb`e53ec000 C:\Windows\system32\USER32.dll
ModLoad: 000007fb`e5510000 000007fb`e5650000 C:\Windows\system32\GDI32.dll
ModLoad: 000007fb`e10e0000 000007fb`e1176000 C:\Windows\SYSTEM32\SHCORE.dll
ModLoad: 000007fb`e35f0000 000007fb`e3640000 C:\Windows\system32\SHLWAPI.dll
ModLoad: 000007fb`e3e30000 000007fb`e5115000 C:\Windows\system32\SHELL32.dll
ModLoad: 000007fb`e1810000 000007fb`e18f3000 C:\Windows\SYSTEM32\UxTheme.dll
0:030>
Ln 0, Col 0 Sys 0:<Local> Proc 000:69c Thrd 030:6e8 ASM OVR CAPS NUM
11:41 AM
11/9/2015


root@vmm: ~
root@vmm:~# uname -a
Linux vmm 3.13.0-62-generic #102-Ubuntu SMP Tue Aug 11 14:29:36 UTC 2015 x86_64 x86_64 GNU/Linux
root@vmm:~# cat /sys/kernel/mm/ksm/run
Ubuntu_14_04 Virtual Machine
user@user-virtual-machine: ~/svn/vmap
* [ATTACK - CREATE PAGES] mapped 1st page to memory (0x7f791c979000)
* [ATTACK - CREATE PAGES] mapped page buffer (0x7f791b6c9000)
* [ATTACK - RUN - FILTERING] filtering rounds are completed, remai
* [ATTACK - RUN - FILTERING] total attack time so far 720 s / 12 m
* [ATTACK - RUN - VERIFICATION] recreating 3527 attack pages
* [ATTACK - CREATE PAGES] win64_server_2012.create_attack_pages()
* [ATTACK - CREATE PAGES] unmap previous buffer
* [ATTACK - CREATE PAGES] 1st page file dump opened (bin/win2012/w
* [ATTACK - CREATE PAGES] mapped 1st page to memory (0x7f791c979000)
* [ATTACK - CREATE PAGES] mapped page buffer (0x7f791ad8f000)
* [ATTACK - RUN - VERIFICATION] start verification rounds (total of 16)
* [ATTACK - RUN - VERIFICATION] wait for pages to be merged (approx. 12 mi
* [ATTACK - RUN - VERIFICATION] verification round 1 done
000007FBE59F0000,
* [ATTACK - RUN - RESULTS] *** HIT: 000007FBE59F0000, rating: 2/2 (address
* [ATTACK SUMMARY]
> ATTACK TIME 1440 s / 24 min
> HITS 1
> FILTERING ROUNDS 1
> VERIFICATION ROUNDS 1
> TOTAL ROUNDS 2
* [done]
user@user-virtual-machine:~/svn/vmap$
Windows_2012_x64 Virtual Machine
Pid 1692 - WinDbg:6.3.9600.17298 AMD64
File Edit View Debug Window Help
Command
Microsoft (R) Windows Debugger Version 6.3.9600.17298 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.
000007fb`e59f0000
Executable search path is:
ModLoad: 000007f7`0a7a0000 000007f7`0a9e3000 C:\Windows\Explorer.EXE
ModLoad: 000007fb`e59f0000 000007fb`e5bae000 C:\Windows\SYSTEM32\ntdll.dll
ModLoad: 000007fb`e3330000 000007fb`e3466000 C:\Windows\system32\KERNEL32.DLL
fb`e2d30000 000007fb`e2e23000 C:\Windows\system32\KERNELBASE.dll
fb`e3540000 000007fb`e35e5000 C:\Windows\system32\msvcrt.dll
fb`e3470000 000007fb`e3533000 C:\Windows\system32\OLEAUT32.dll
fb`e3910000 000007fb`e3ac0000 C:\Windows\SYSTEM32\combase.dll
fb`e28d0000 000007fb`e2913000 C:\Windows\SYSTEM32\powrprof.dll
fb`e5910000 000007fb`e59ee000 C:\Windows\SYSTEM32\advapi32.dll
fb`e52a0000 000007fb`e53ec000 C:\Windows\system32\USER32.dll
fb`e5510000 000007fb`e5650000 C:\Windows\system32\GDI32.dll
fb`e10e0000 000007fb`e1176000 C:\Windows\SYSTEM32\SHCORE.dll
ModLoad: 000007fb`e35f0000 000007fb`e3640000 C:\Windows\system32\SHLWAPI.dll
ModLoad: 000007fb`e3e30000 000007fb`e5115000 C:\Windows\system32\SHELL32.dll
ModLoad: 000007fb`e1810000 000007fb`e18f3000 C:\Windows\SYSTEM32\UxTheme.dll
0:030>
Ln 0, Col 0 Sys 0:<Local> Proc 000:69c Thrd 030:6e8 ASM OVR CAPS NUM
11:41 AM
11/9/2015

Speed improvements

> Many ways to increase speed of attack

Speed improvements

> Many ways to increase speed of attack

> Allocate more random pages in-between

Speed improvements

> Many ways to increase speed of attack

> Allocate more random pages in-between

> Use more than one guess page (redundancy)

Speed improvements

> Many ways to increase speed of attack

> Allocate more random pages in-between

> Use more than one guess page (redundancy)

> Different guess pages for same secret
e.g. relocated code pages ☺

Big limitation

> No control over victim memory layout

Big limitation

> No control over victim memory layout

> Some control would help a lot ☺

Big limitation

> No control over victim memory layout

    > Some control would help a lot ☺

> No write primitive

Big limitation

> No control over victim memory layout

    > Some control would help a lot ☺

> No write primitive

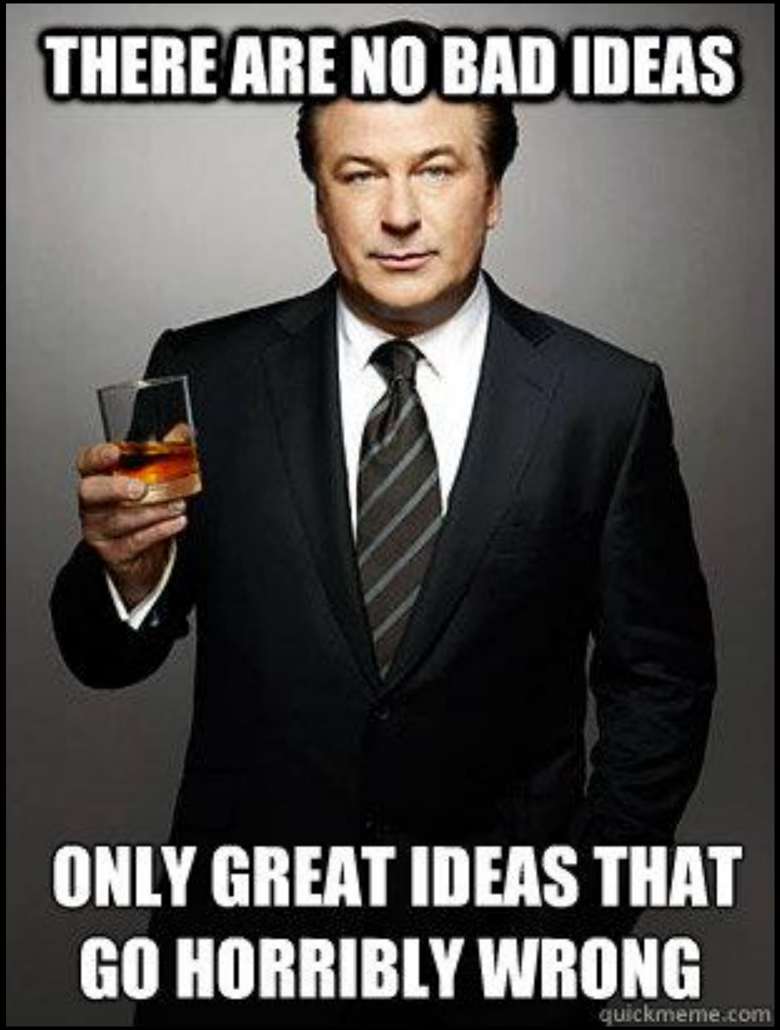
    > Rowhammer ☺

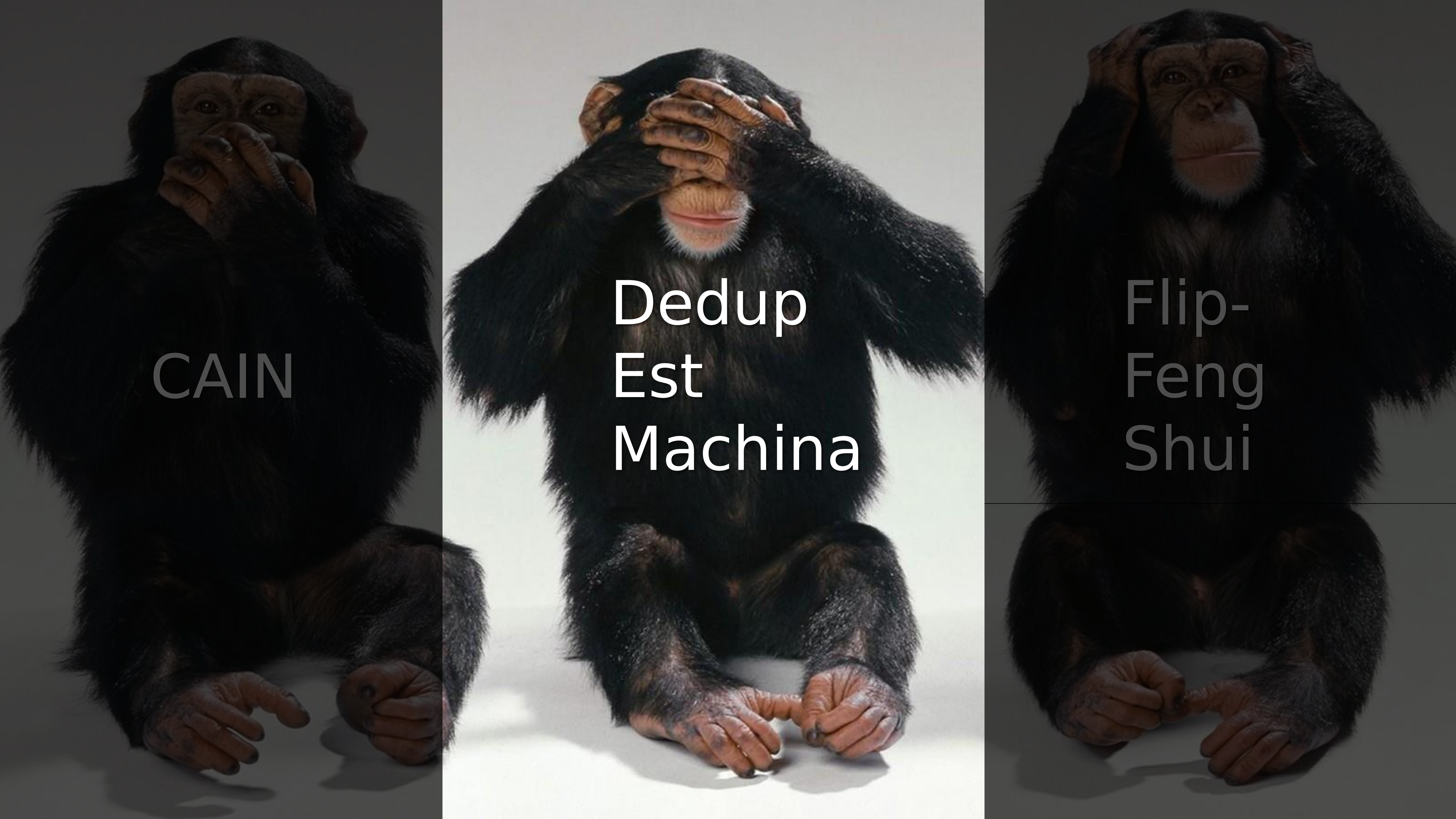
memdedup for Windows

> MS enabled memory deduplication for Windows 8.1 + 10

memdedup for Windows

> MS enabled memory deduplication for Windows 8.1 + 10




CAIN
Dedup Est Machina
Flip-Feng Shui

Dedup est Machina

Deduplication
(software side-channel)

Dedup est Machina

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

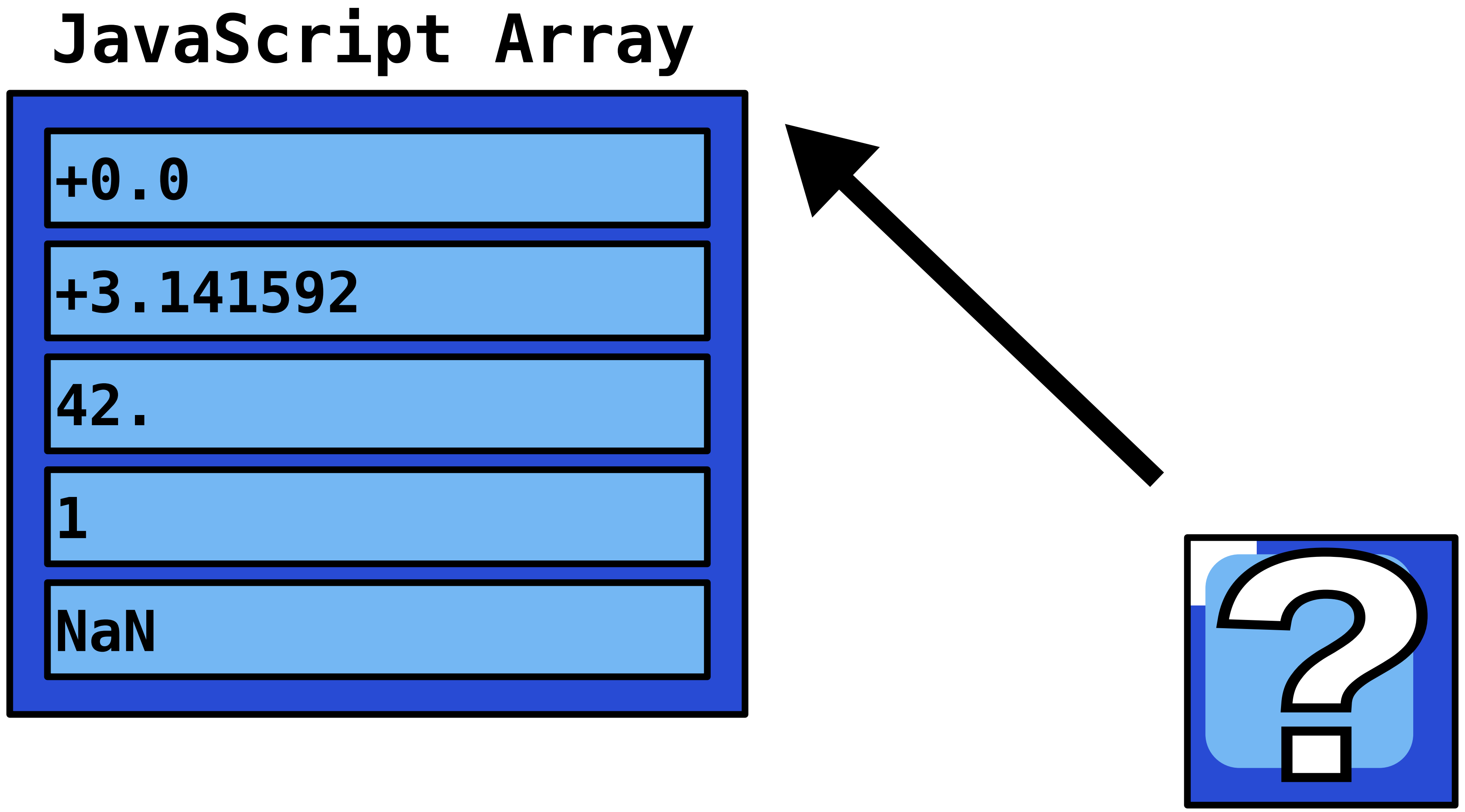
Dedup est Machina

Deduplication
(software side-channel)
+
Rowhammer
(hardware bug)
↓
Exploit MS Edge without software bugs
(from JavaScript)

Outline:

Deduplication

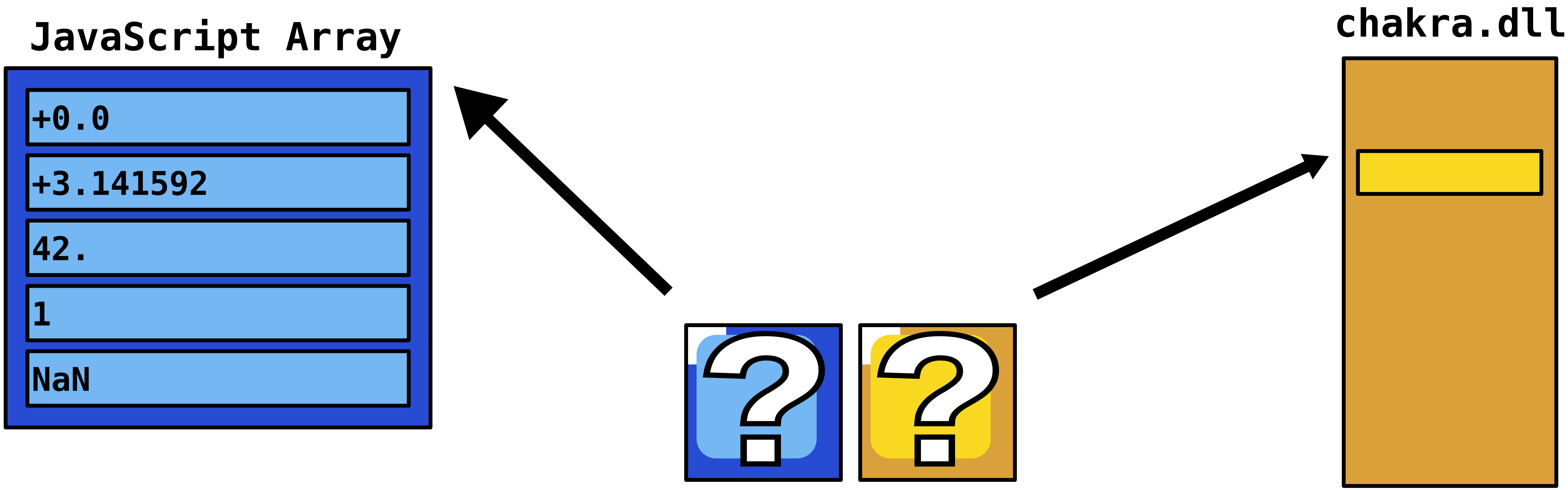
- leak heap & code addresses



Outline:

Deduplication

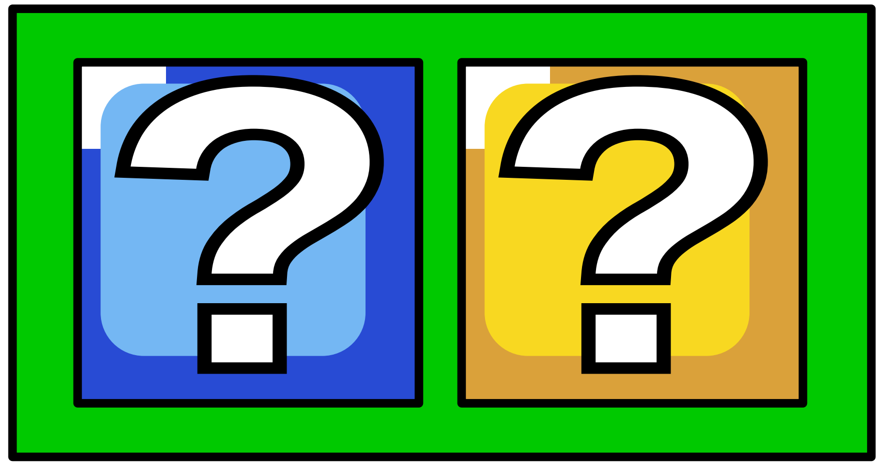
- leak heap & code addresses



Outline:

Deduplication

- leak heap & code addresses
- create a fake object



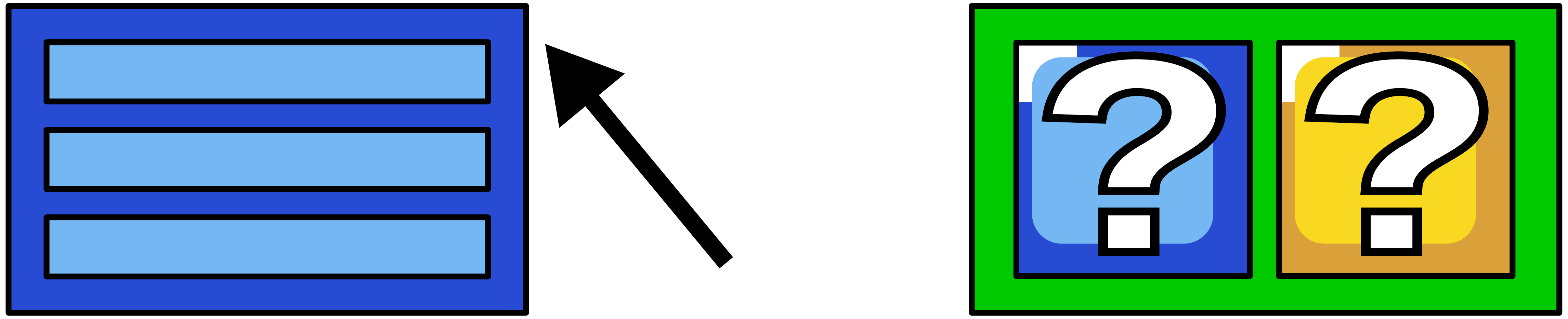
Outline:

Deduplication

- leak heap & code addresses
- create a fake object

Rowhammer

- create reference to our fake object



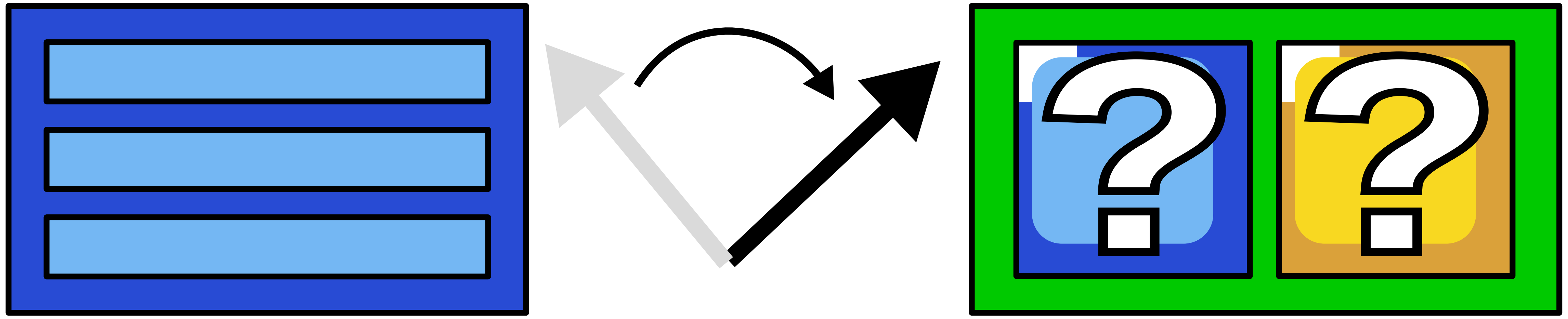
Outline:

Deduplication

- leak heap & code addresses
- create a fake object

Rowhammer

- create reference to our fake object



Leaking existing pages is slow and the gained information is limited.

What if we can manipulate the contents of the victim's memory to leak secrets hand-picked by the attacker.

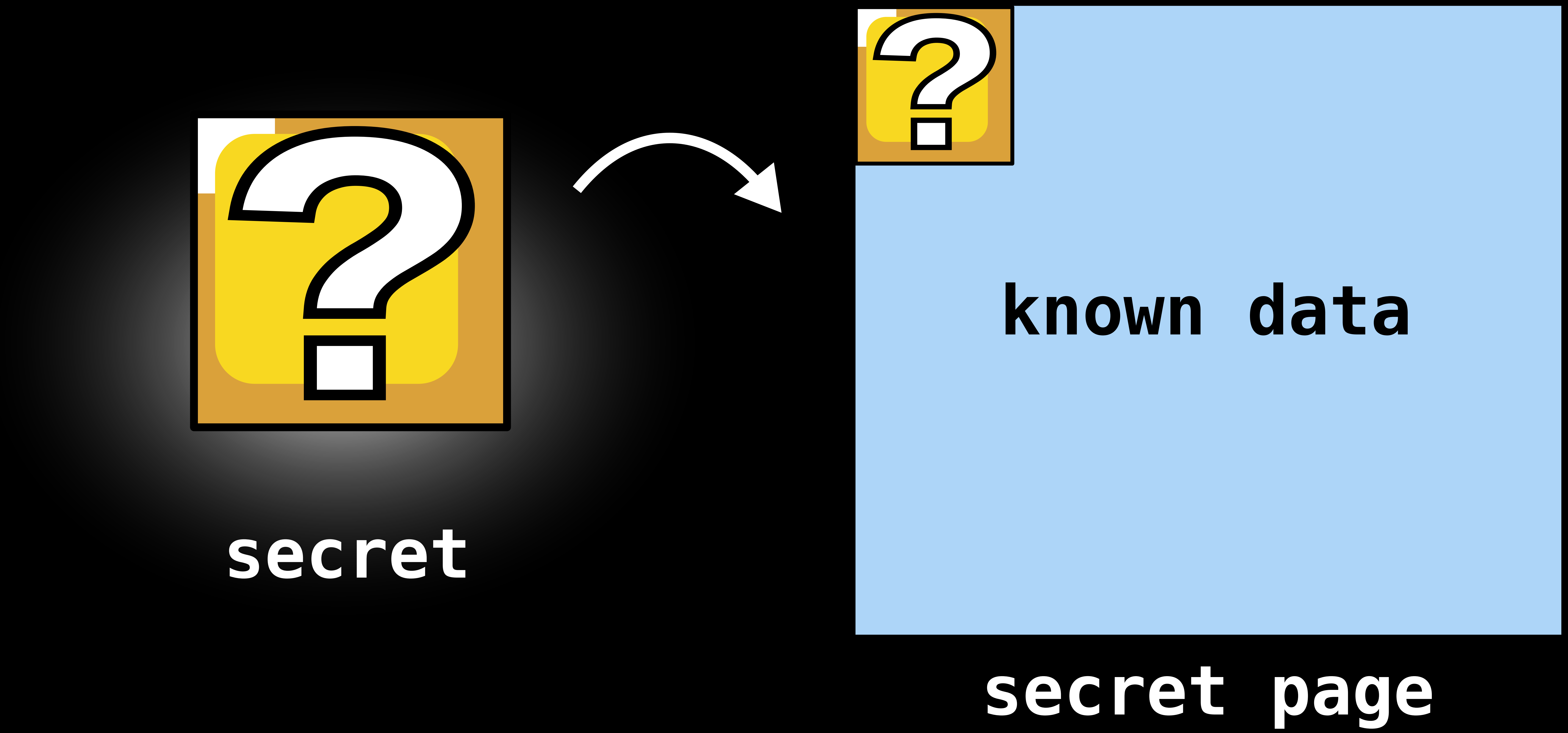
Challenge 1:

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

Turning a secret into a page

secret

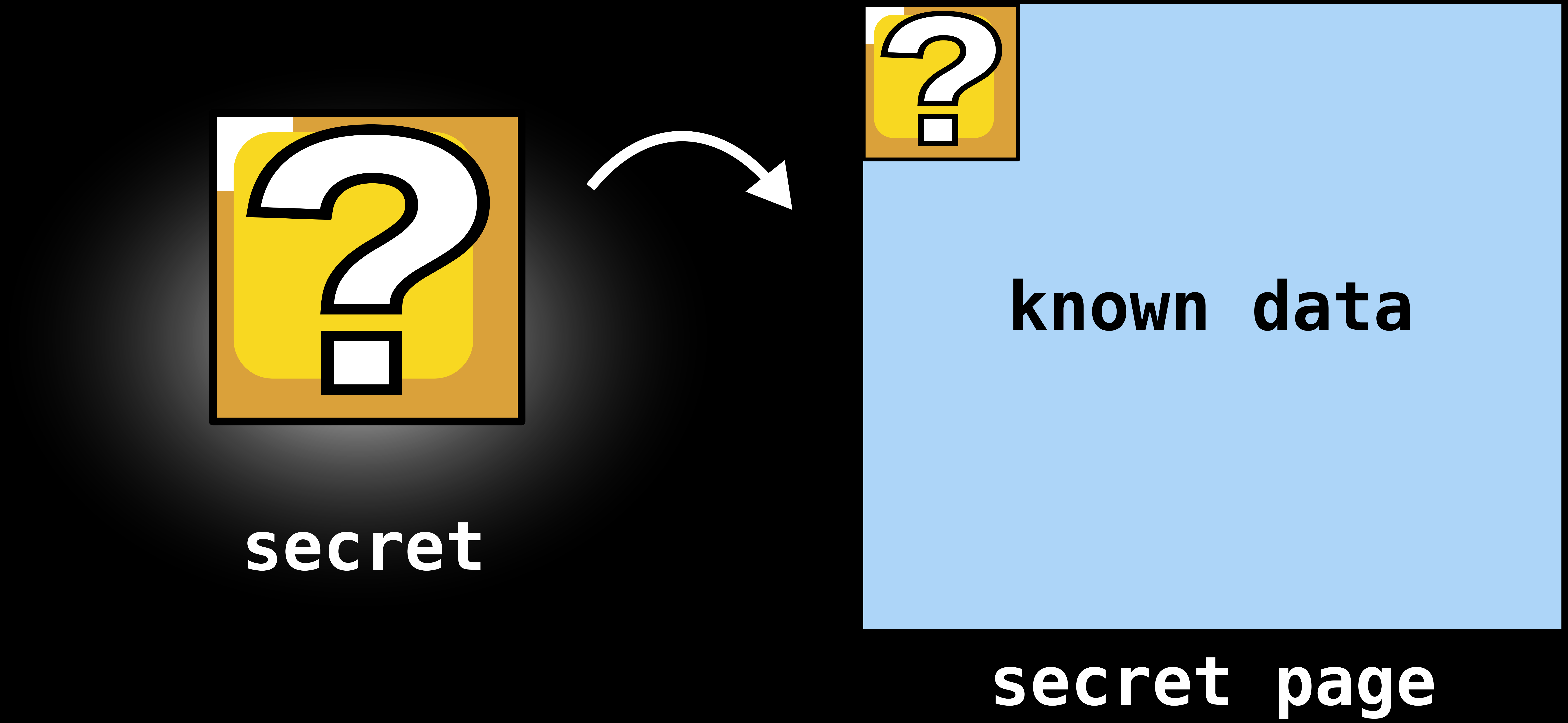
Turning a secret into a page



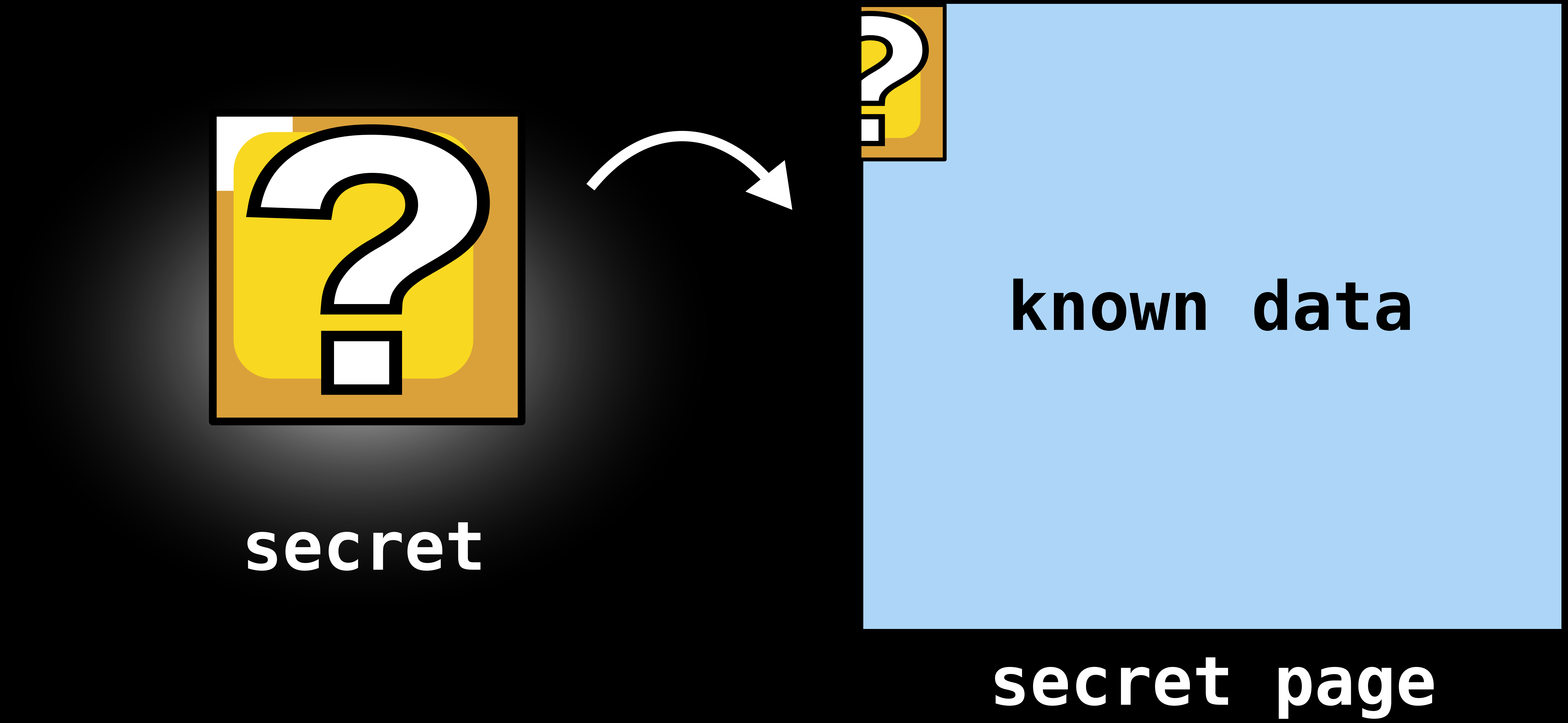
Challenge 2:

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

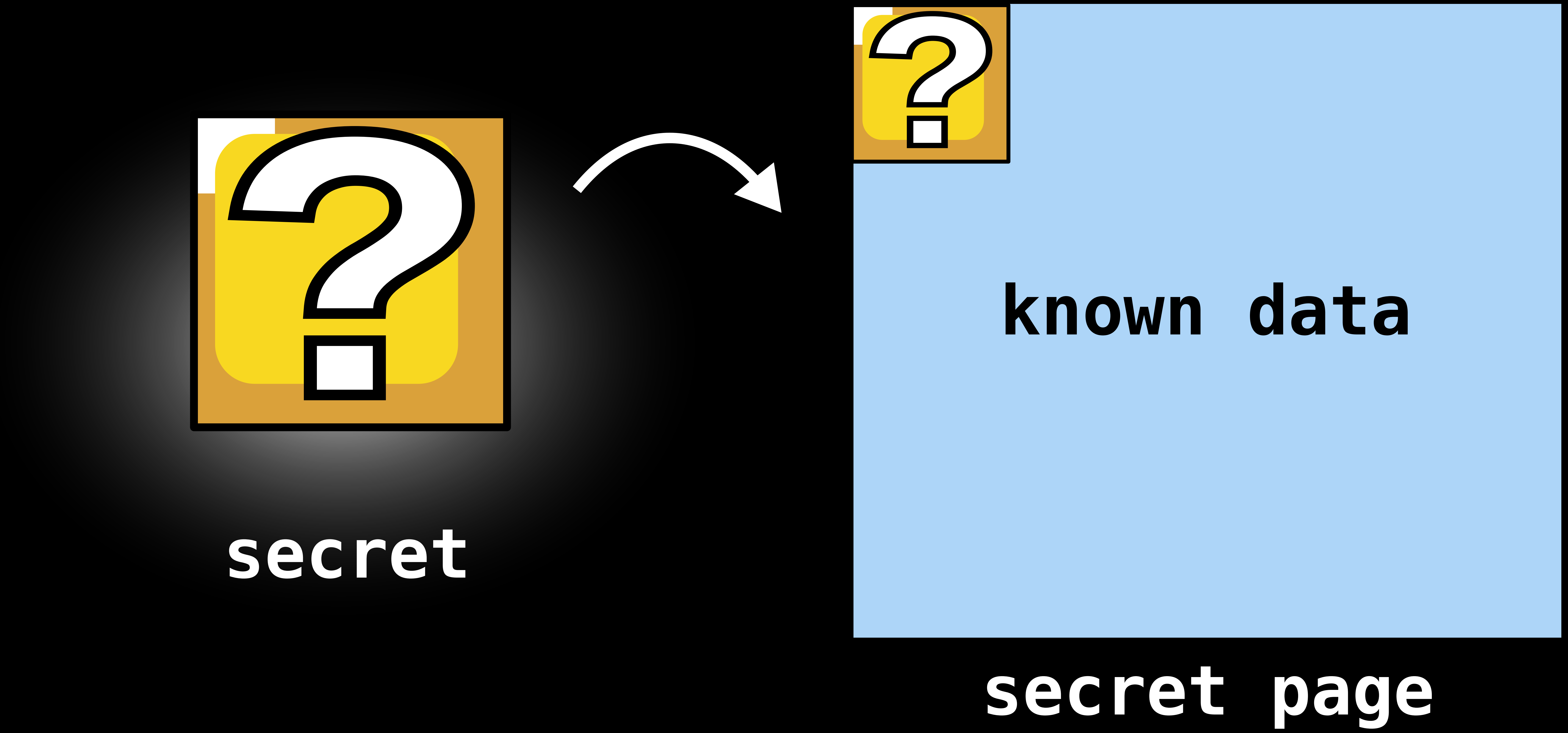
Primitive #1: alignment probing



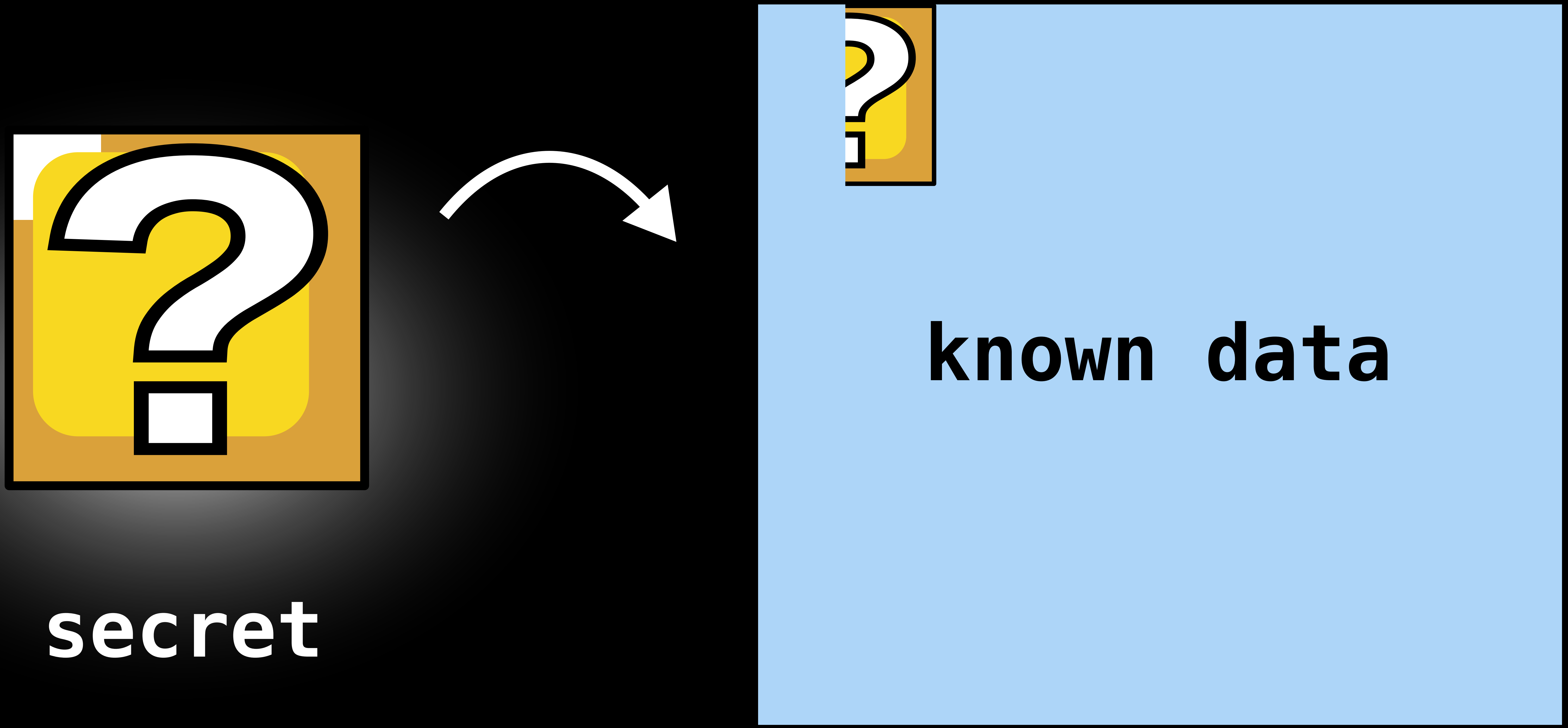
Primitive #1: alignment probing



Primitive #2: partial reuse



Primitive #2: partial reuse



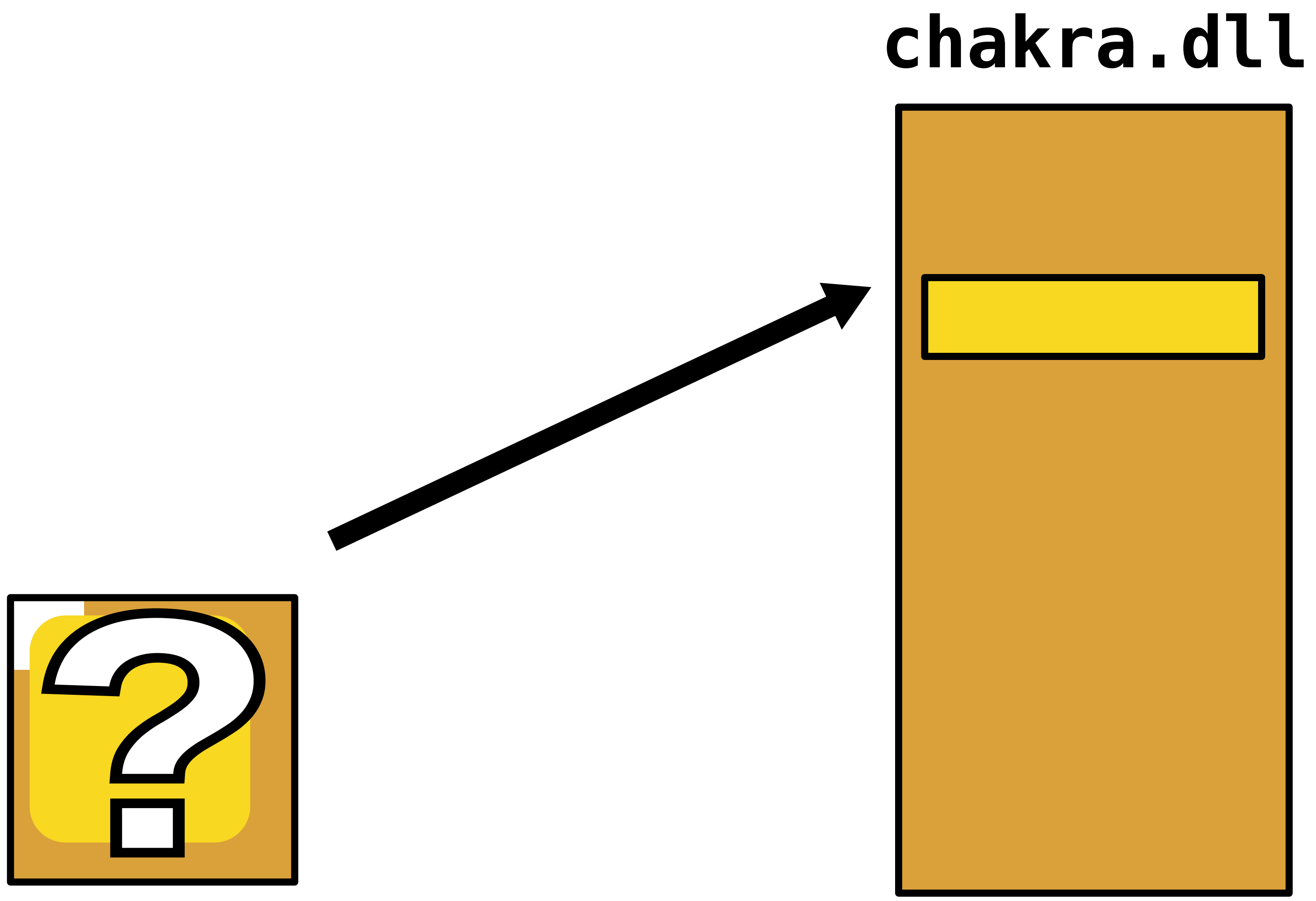
secret

secret page

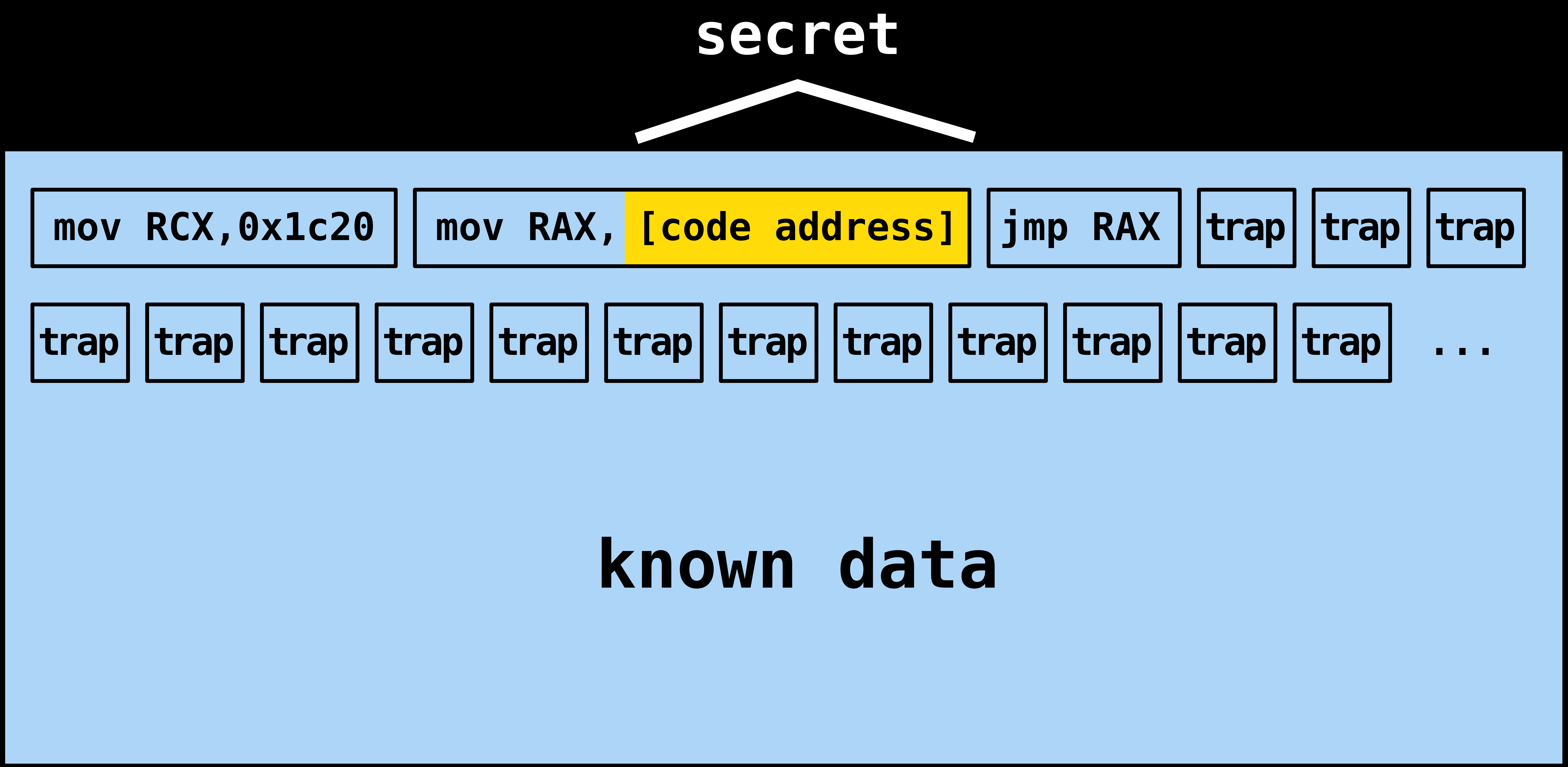
Outline:

Deduplication

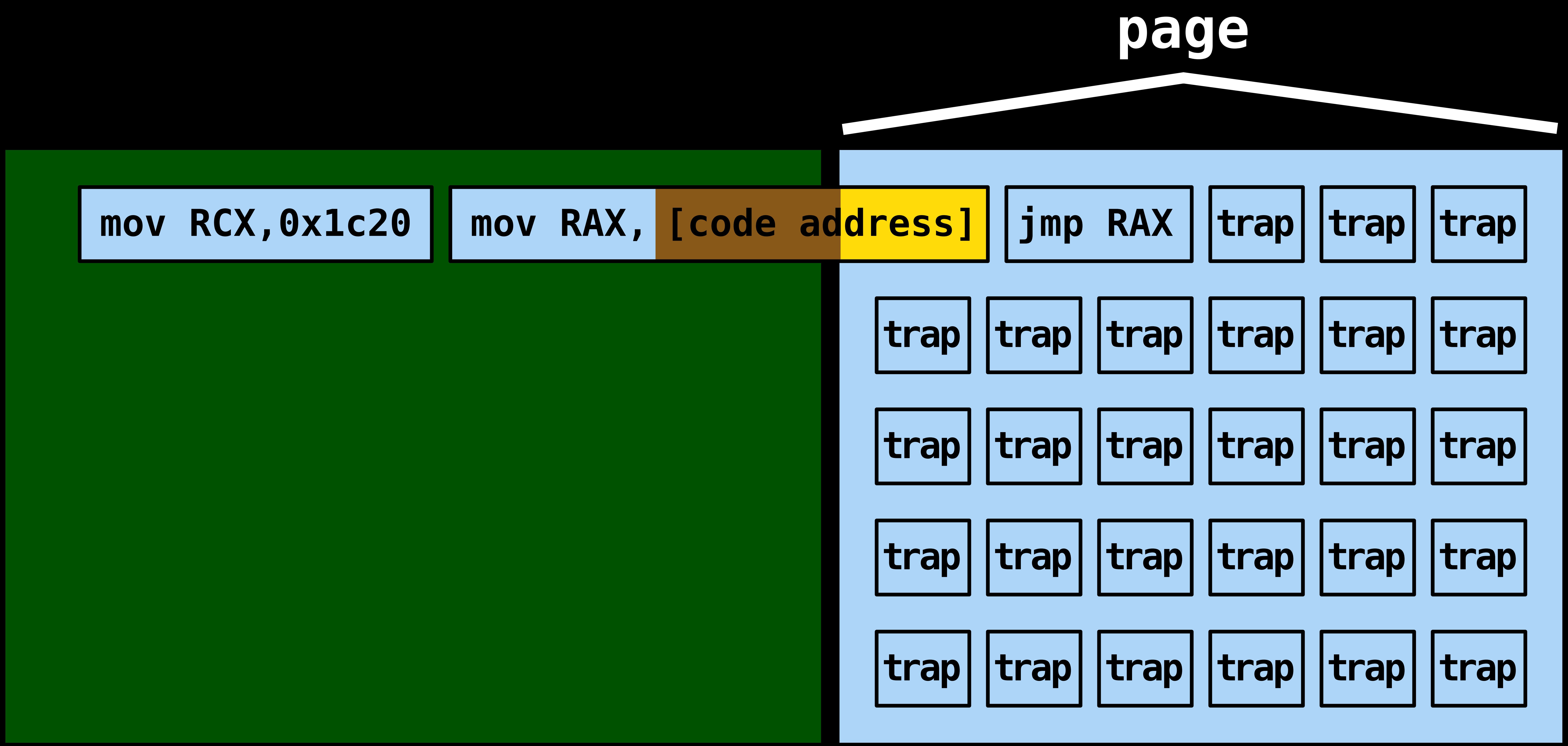
- leak heap & code addresses



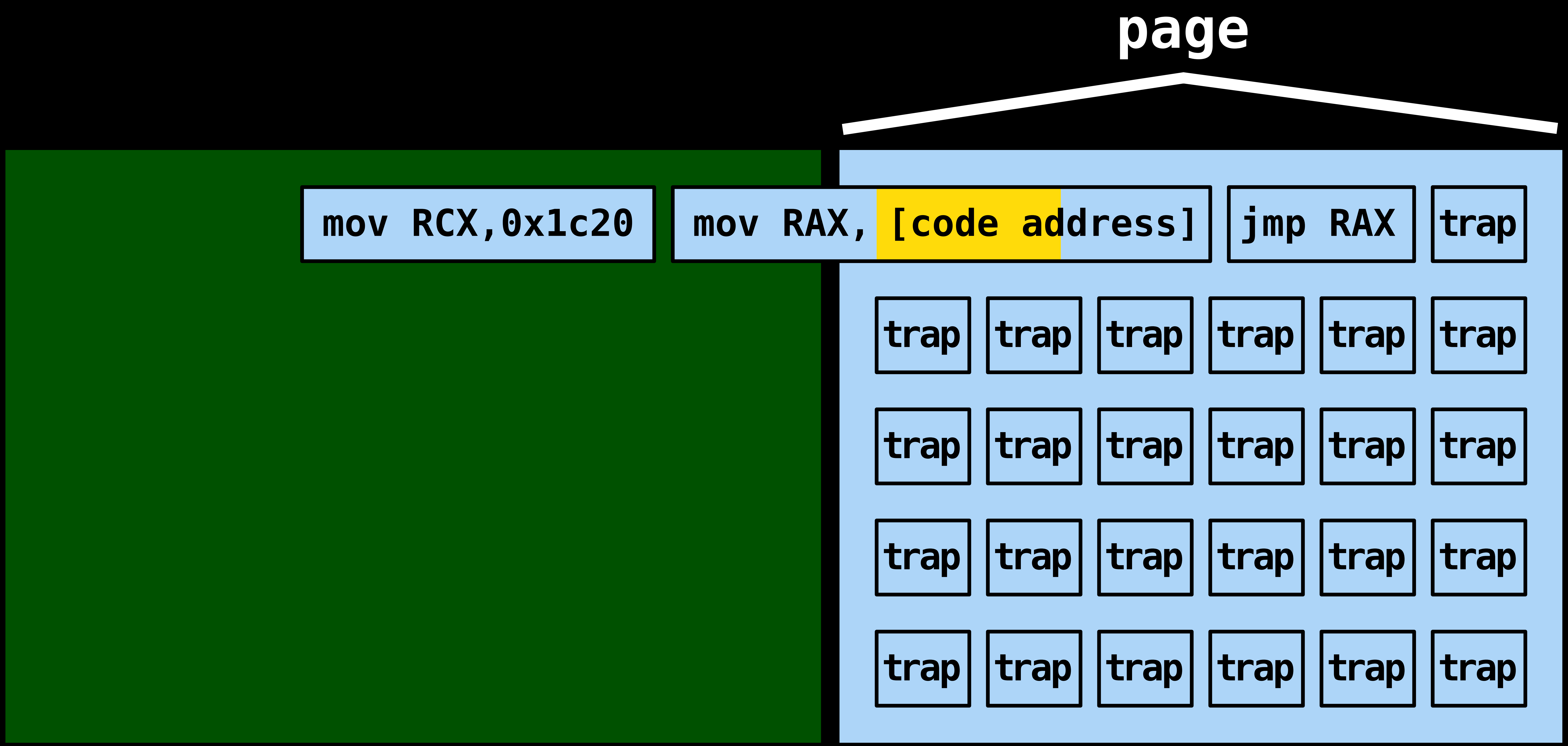
JIT function epilogue (MS Edge)



JIT function epilogue (MS Edge)



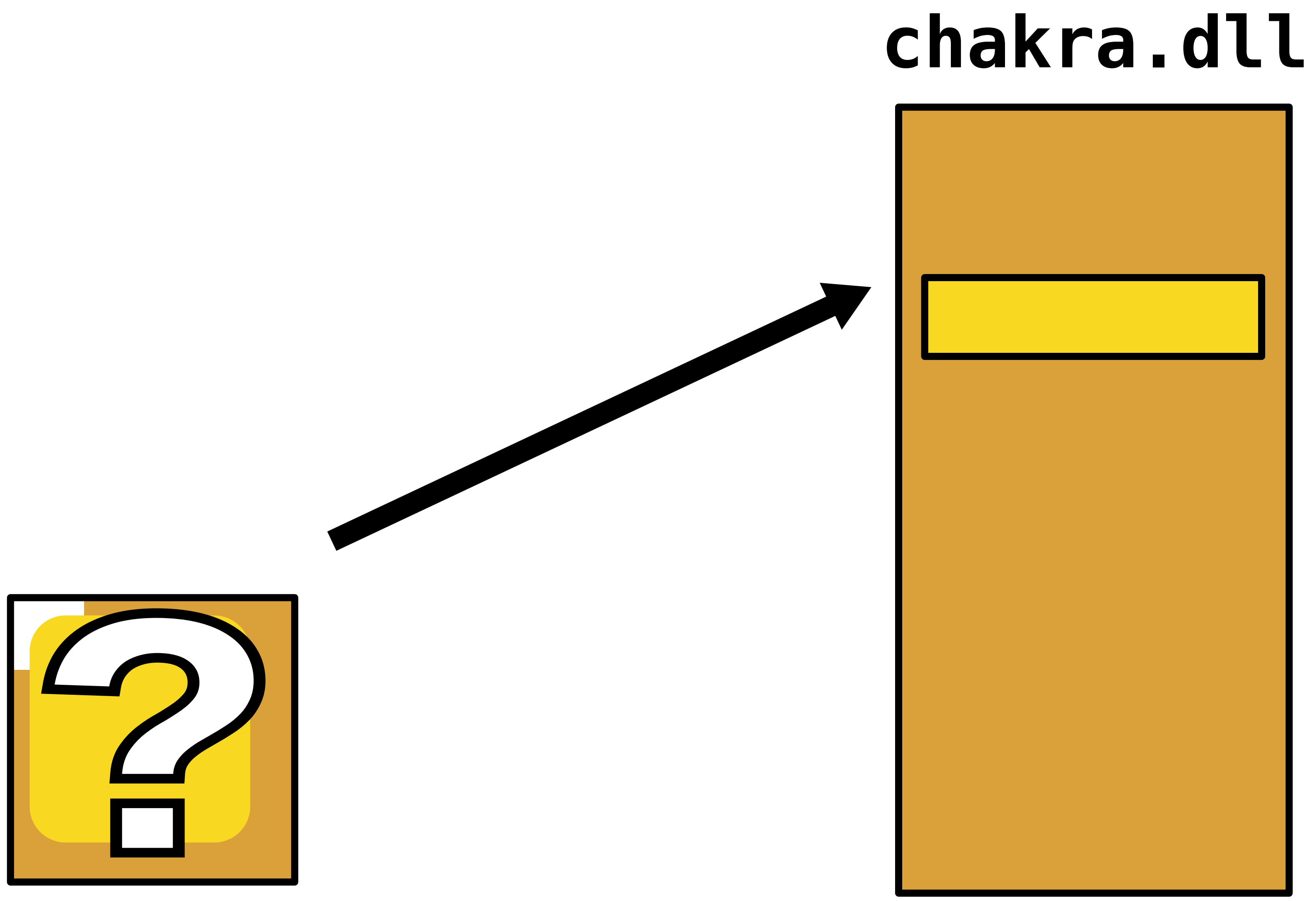
JIT function epilogue (MS Edge)



Outline:

Deduplication

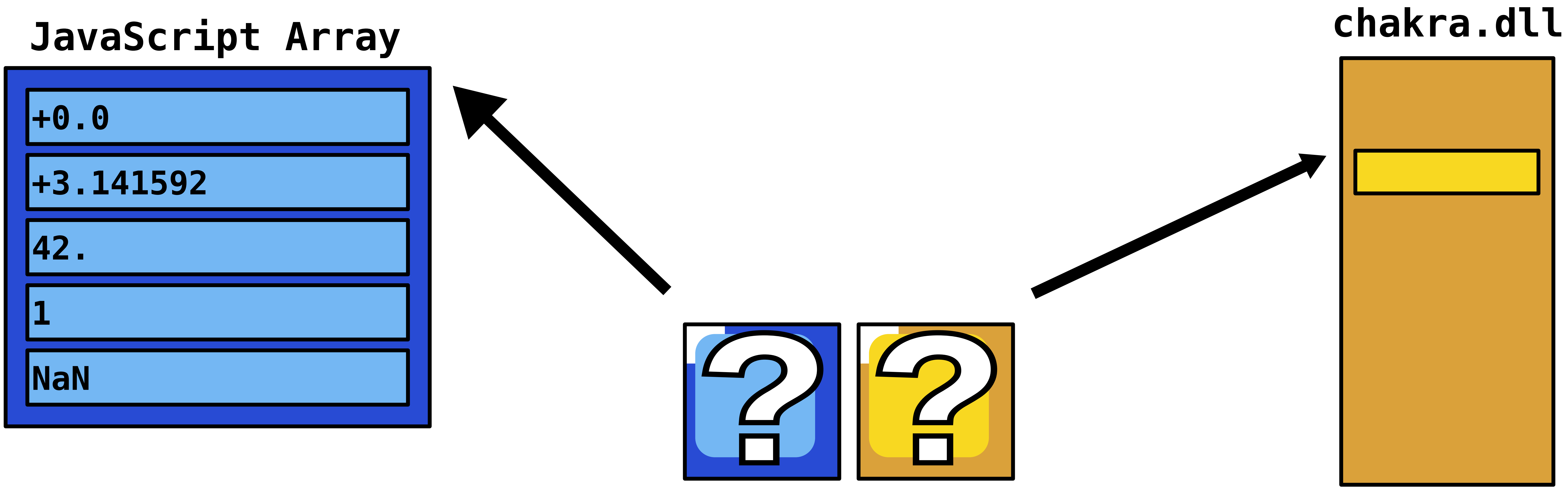
- leak heap & code addresses



Outline:

Deduplication

- leak heap & code addresses



What if leaking a heap pointer in stages is not possible...

We need to guess a page containing the complete pointer.

Heap pointer entropy in Edge

0x5F48143540

Heap pointer entropy in Edge

64G

advertised ASLR (24 bit) * redundancy

$\overbrace{}$

0x5F48143540

Heap pointer entropy in Edge

64G

advertised ASLR (24 bit) * redundancy

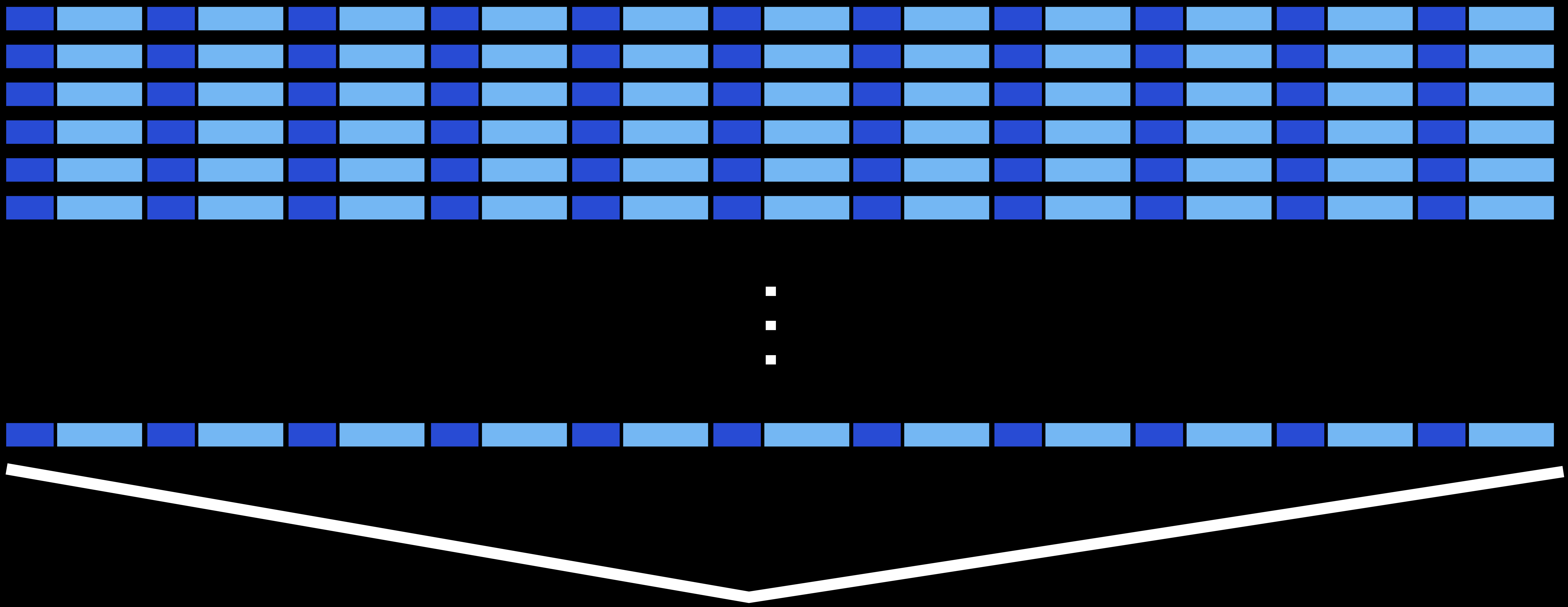
0x5F48143540

256T

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

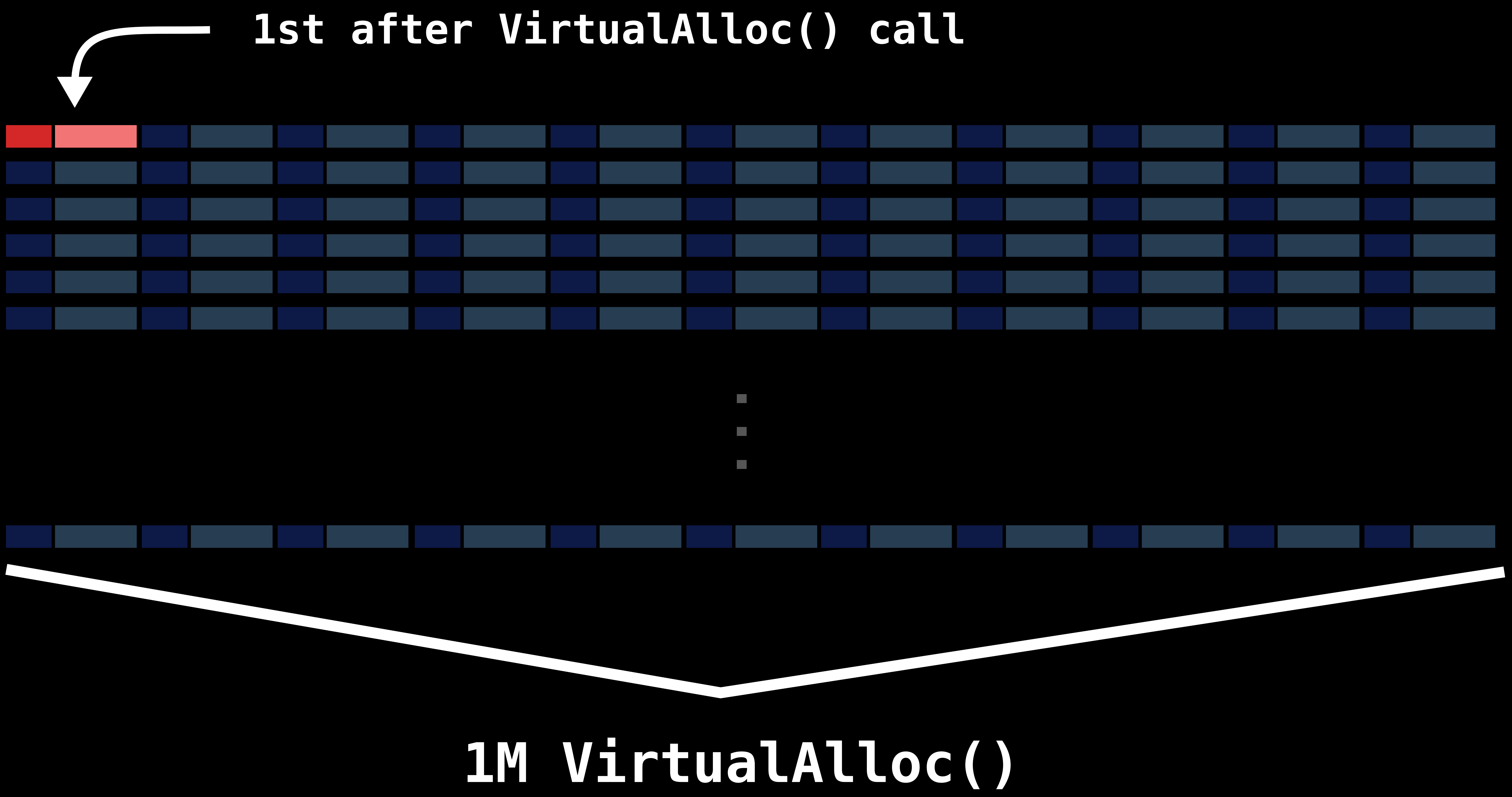
Slab allocator for JavaScript objects

Slab allocator for JavaScript objects

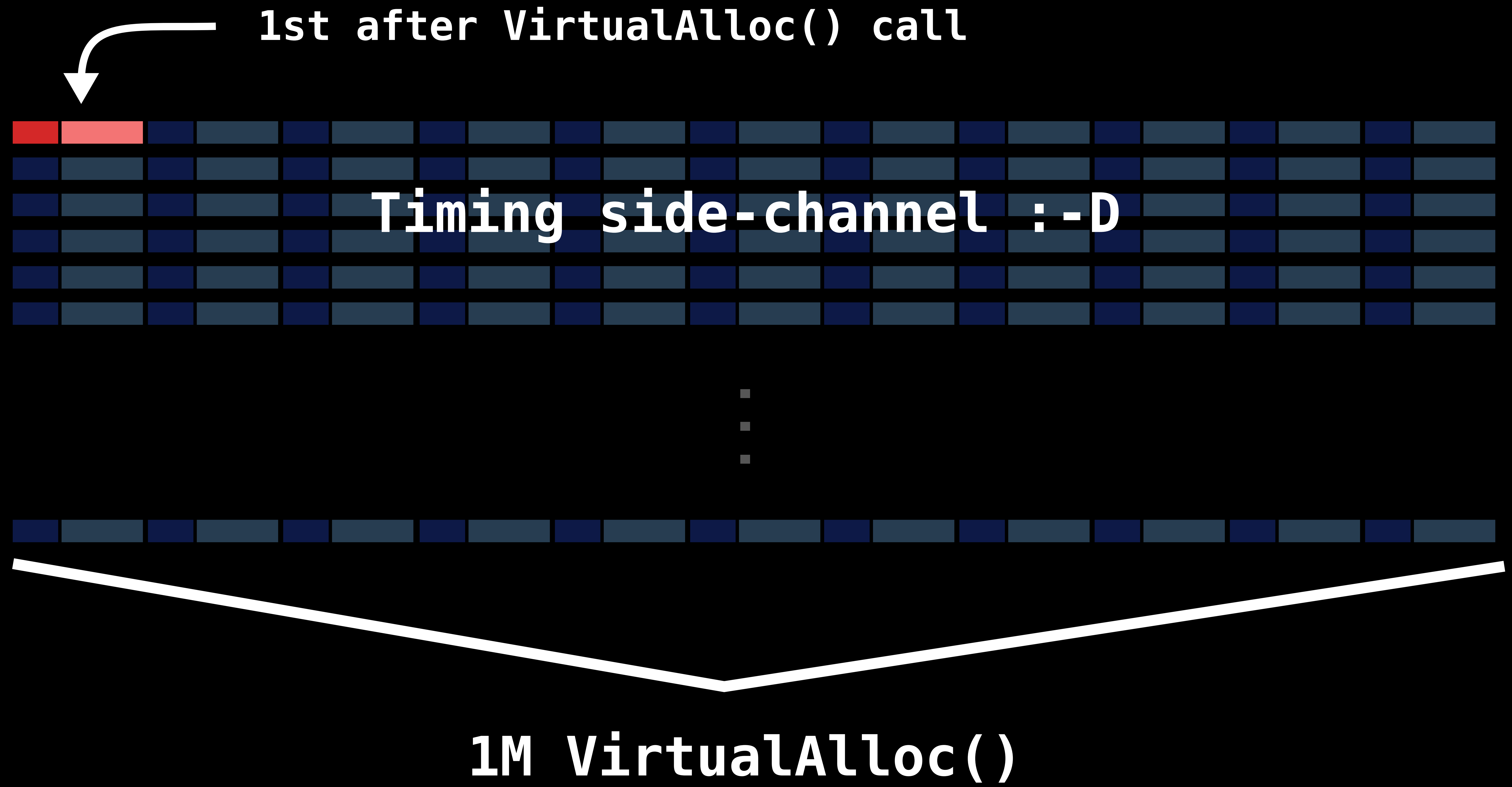


1M VirtualAlloc()

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 * redundancy

advertised ASLR (24 bit)

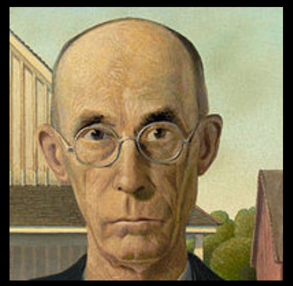
0x5F48100000

4G * redundancy

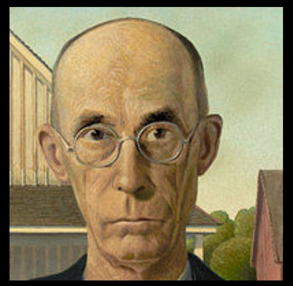
entropy after 1MB alignment (20 bit)

Birthday problem

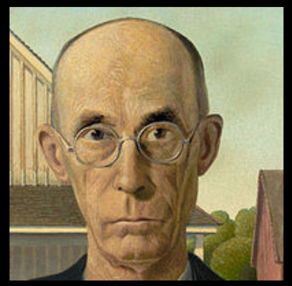
Birthday problem



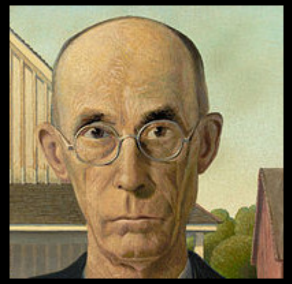
Birthday problem



Birthday problem



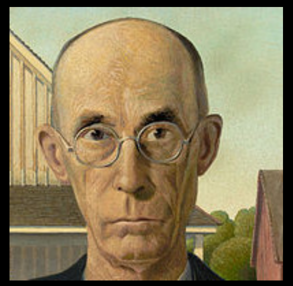
Birthday problem



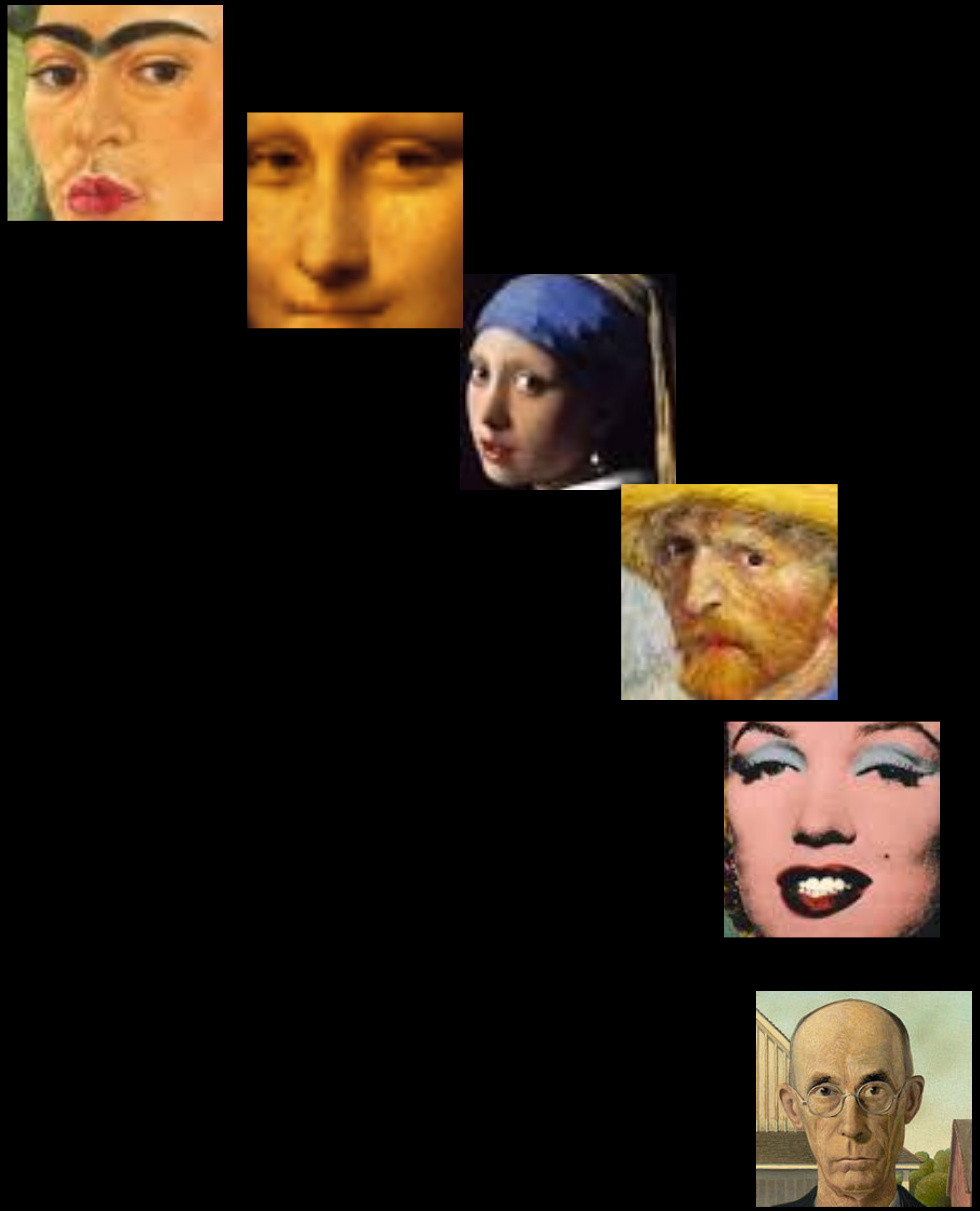
Birthday problem



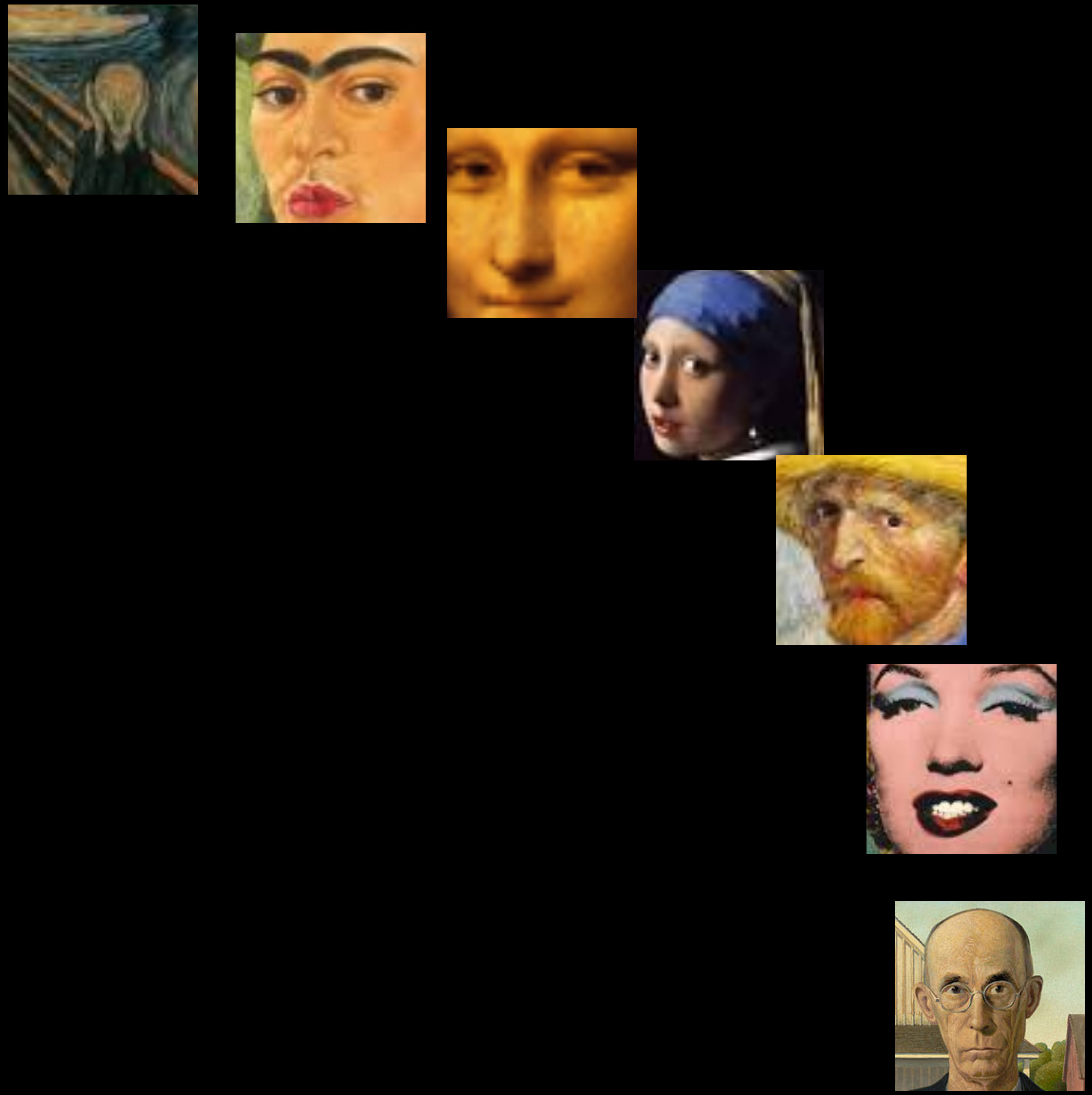




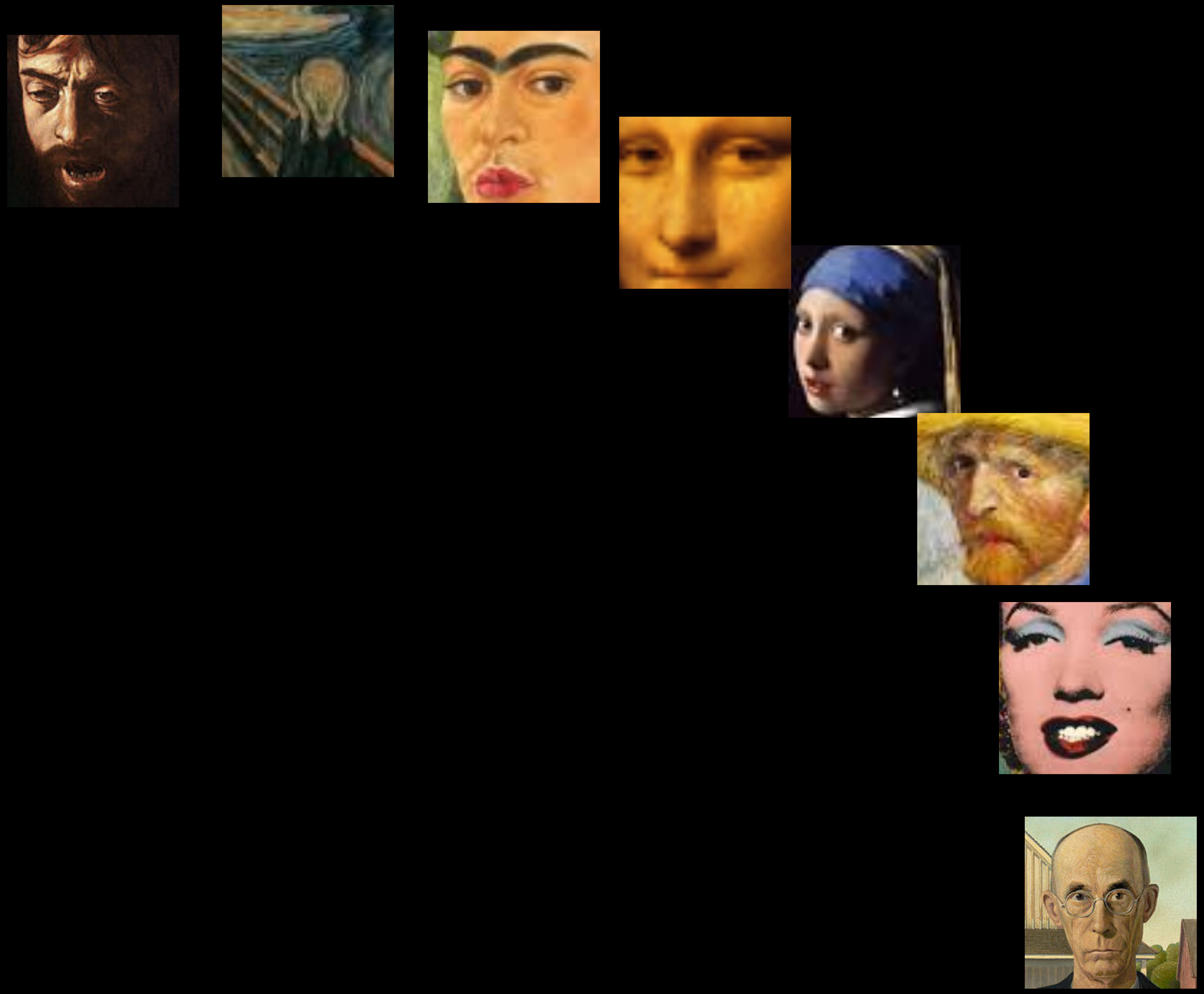
Birthday problem



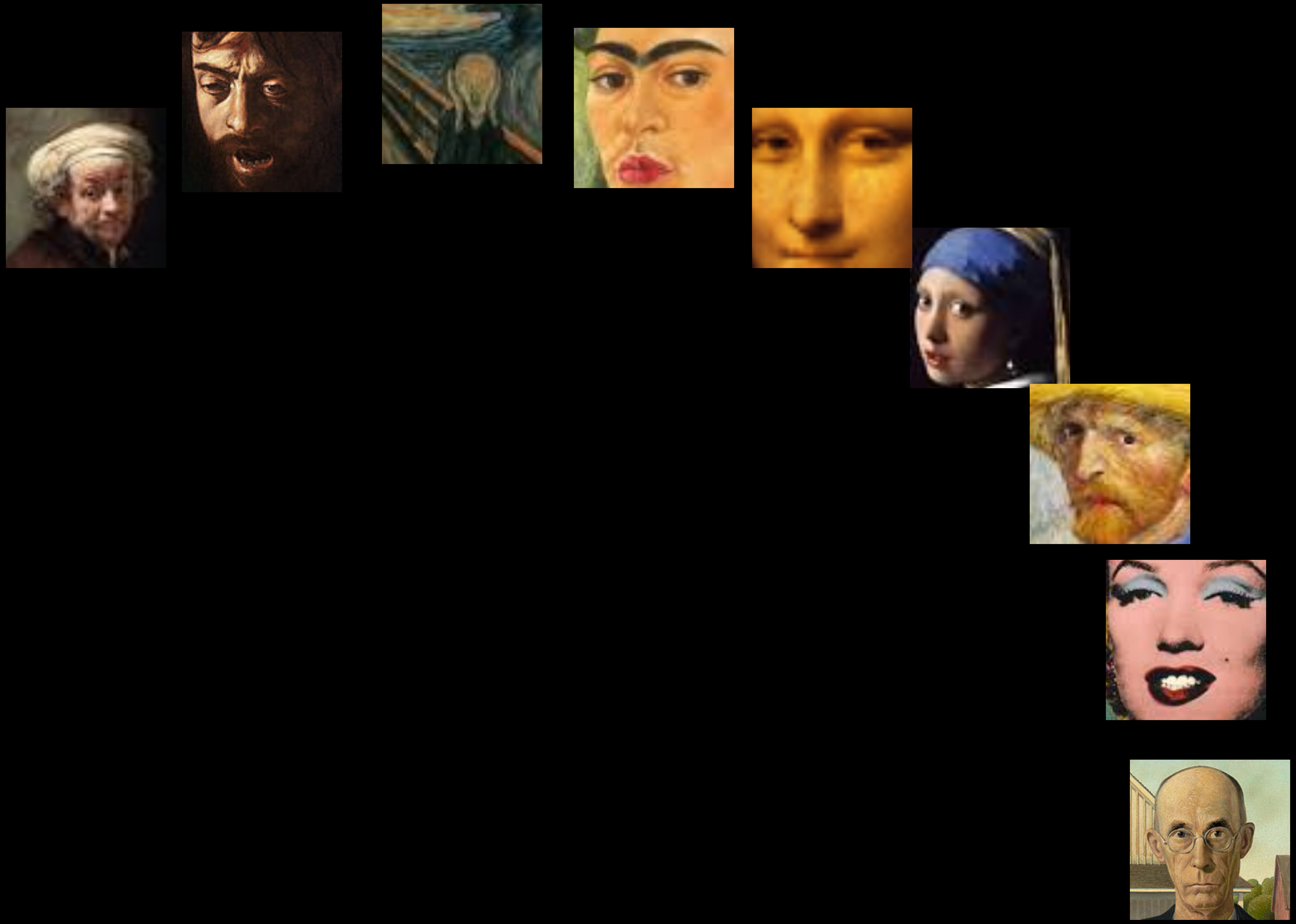
Birthday problem



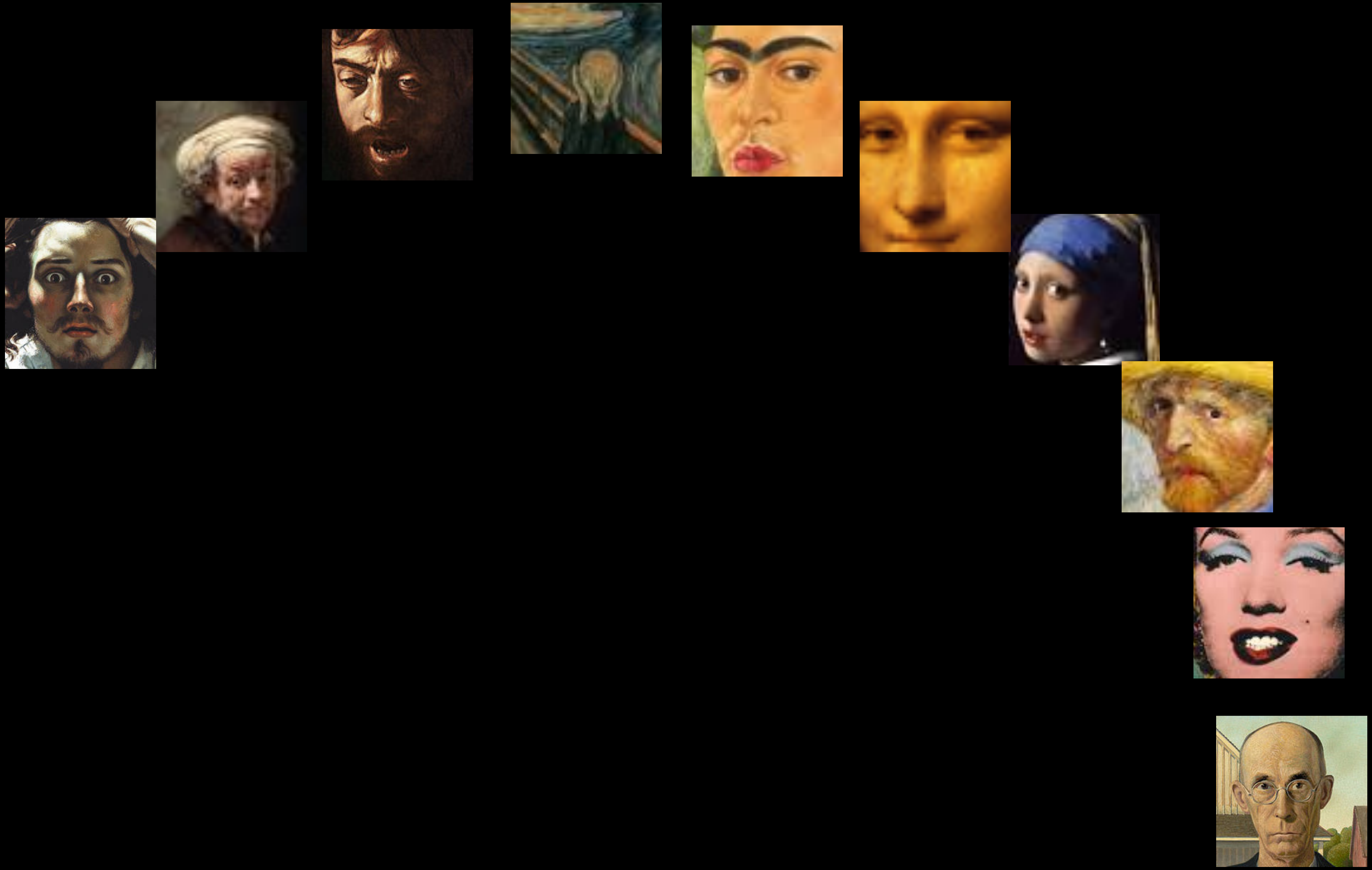
Birthday problem



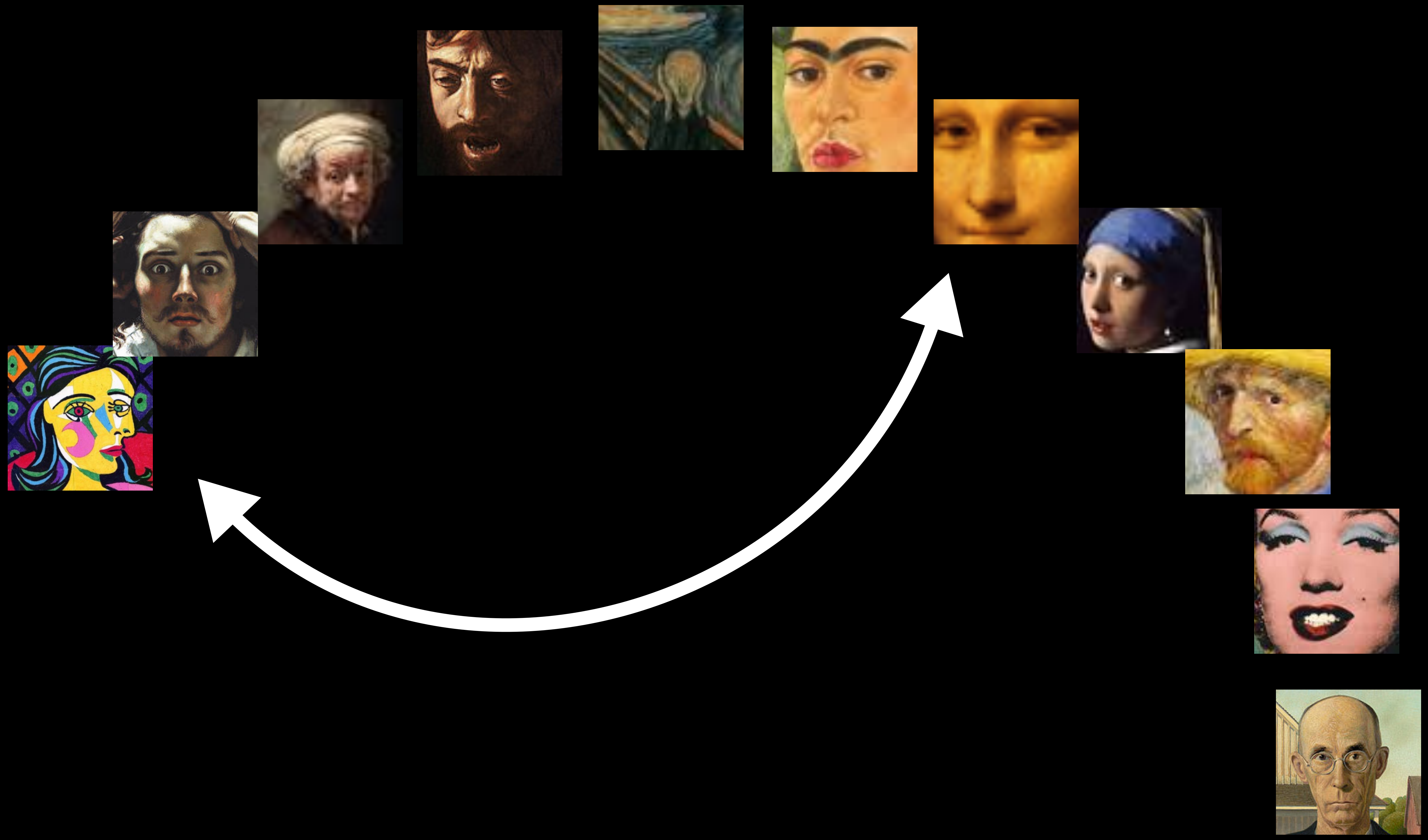
Birthday problem



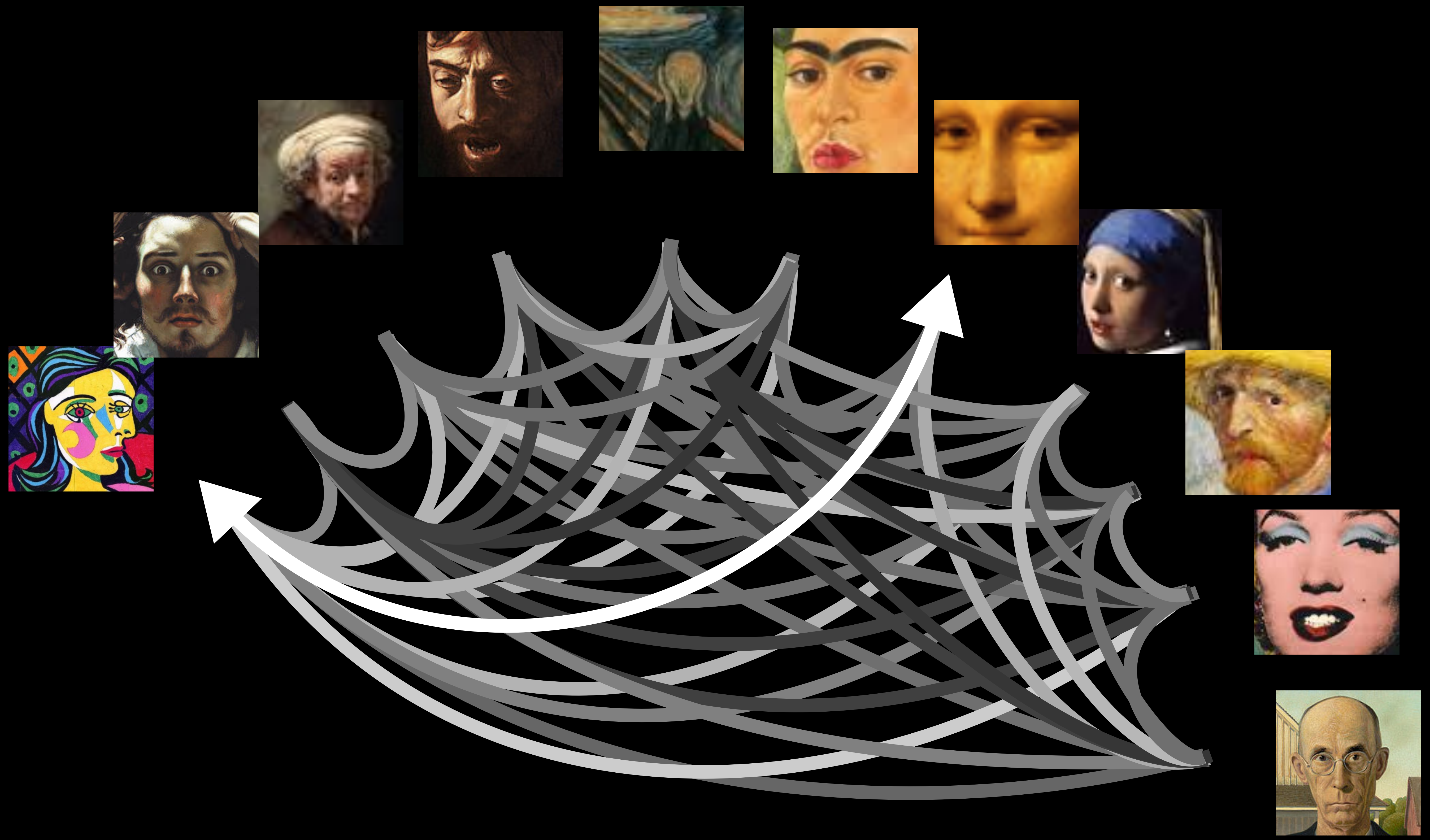
Birthday problem



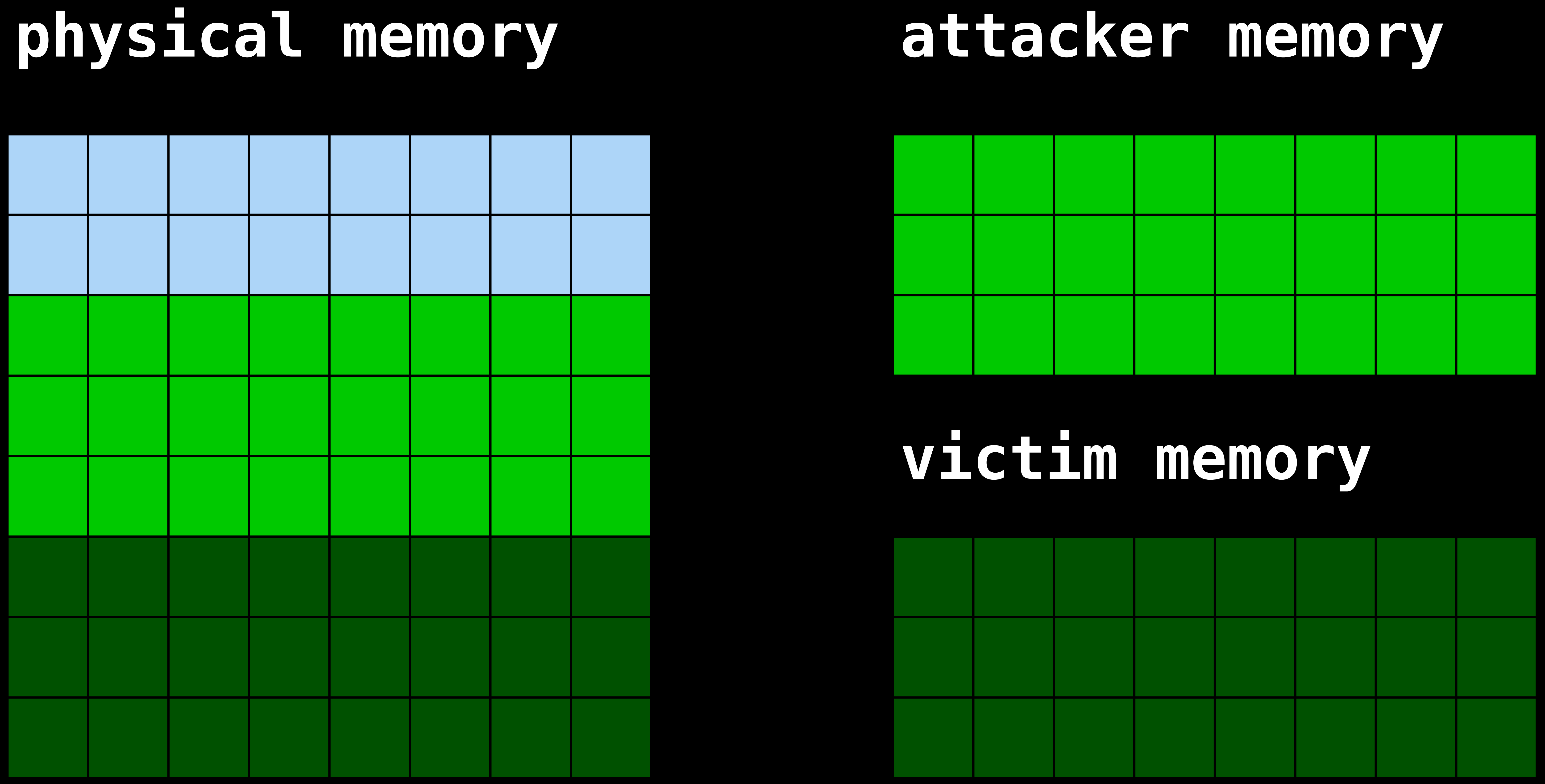
Birthday problem



Birthday problem

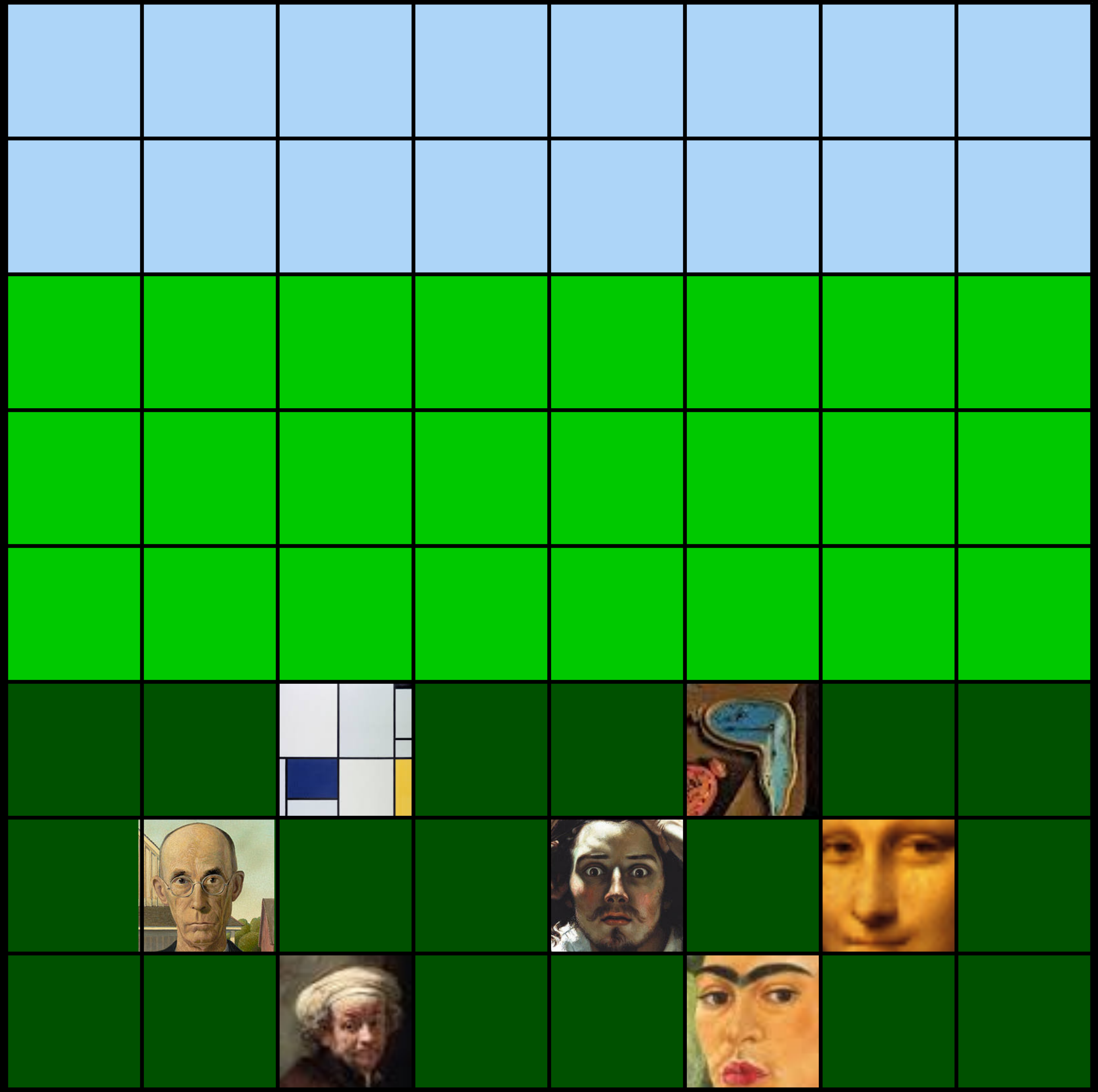


Primitive #3: birthday heapspray

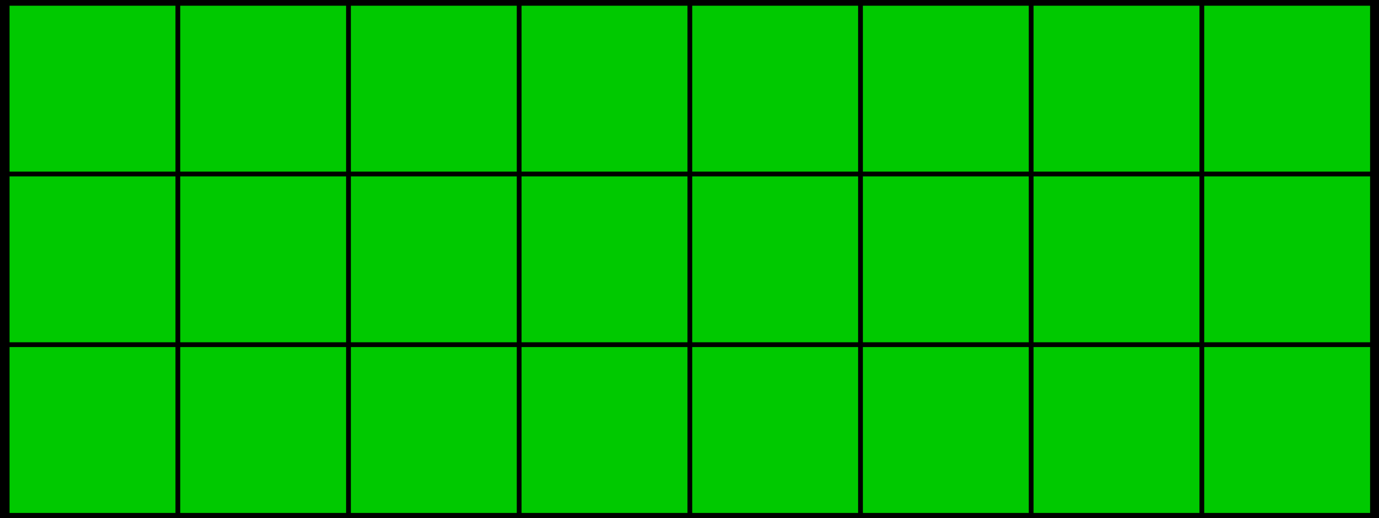


Primitive #3: birthday heapspray

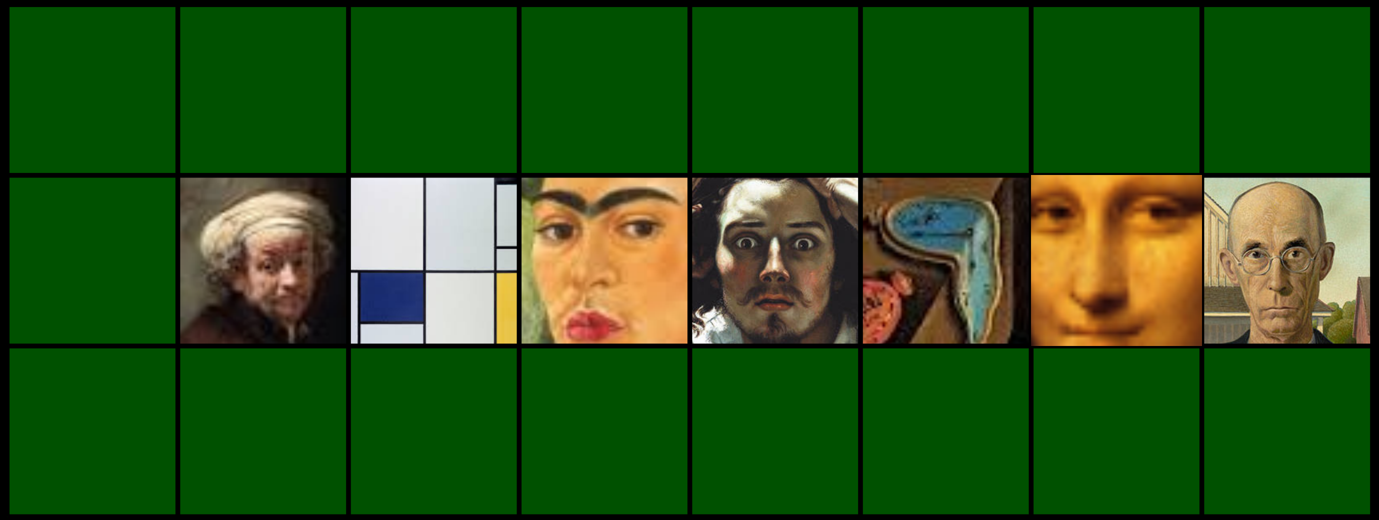
physical memory



attacker memory

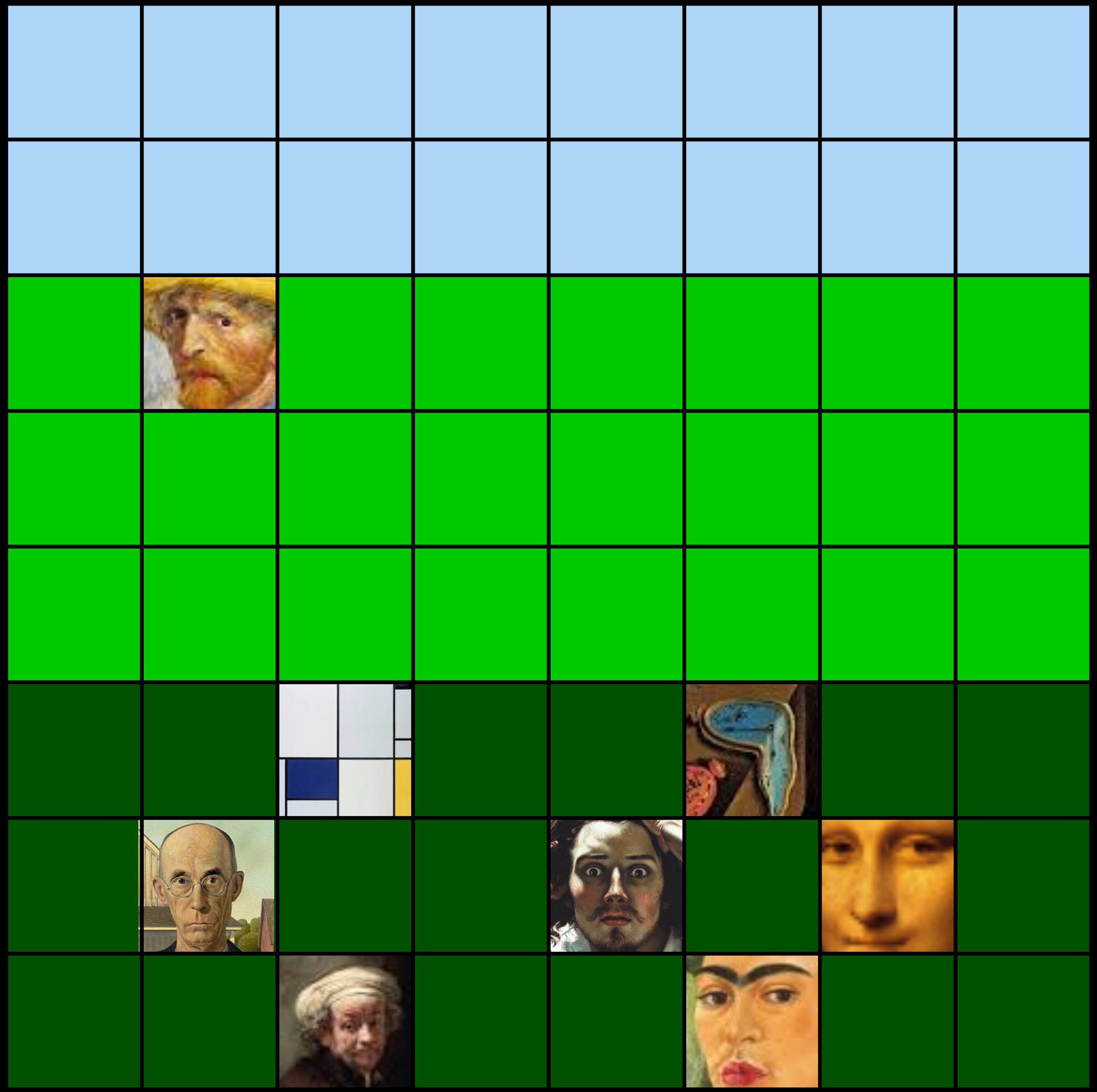


victim memory

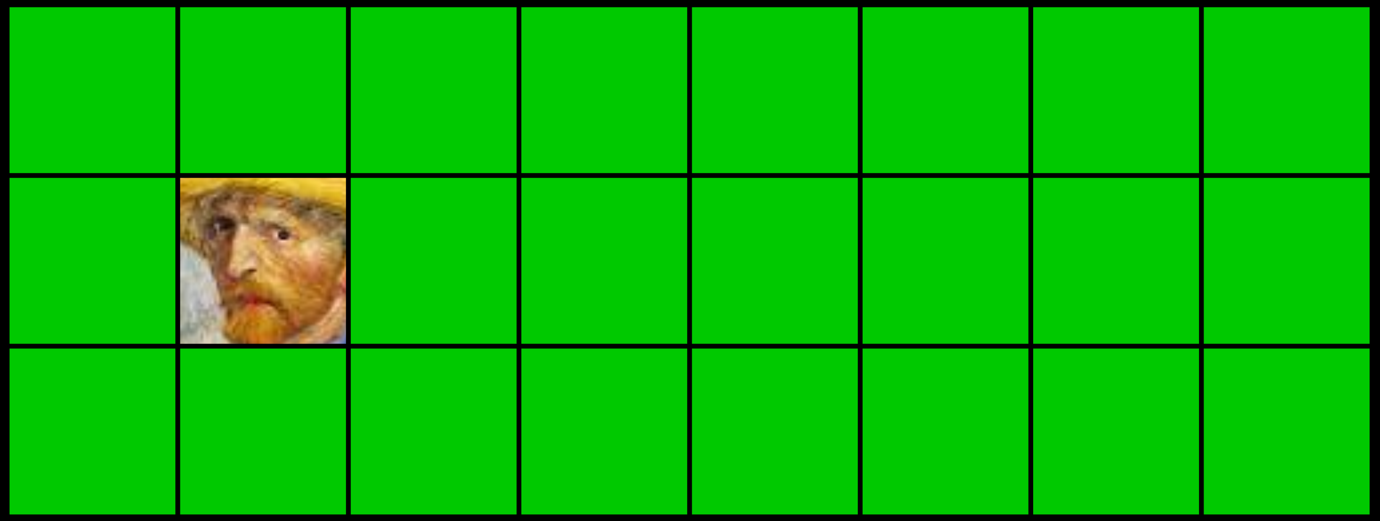


Primitive #3: birthday heapspray

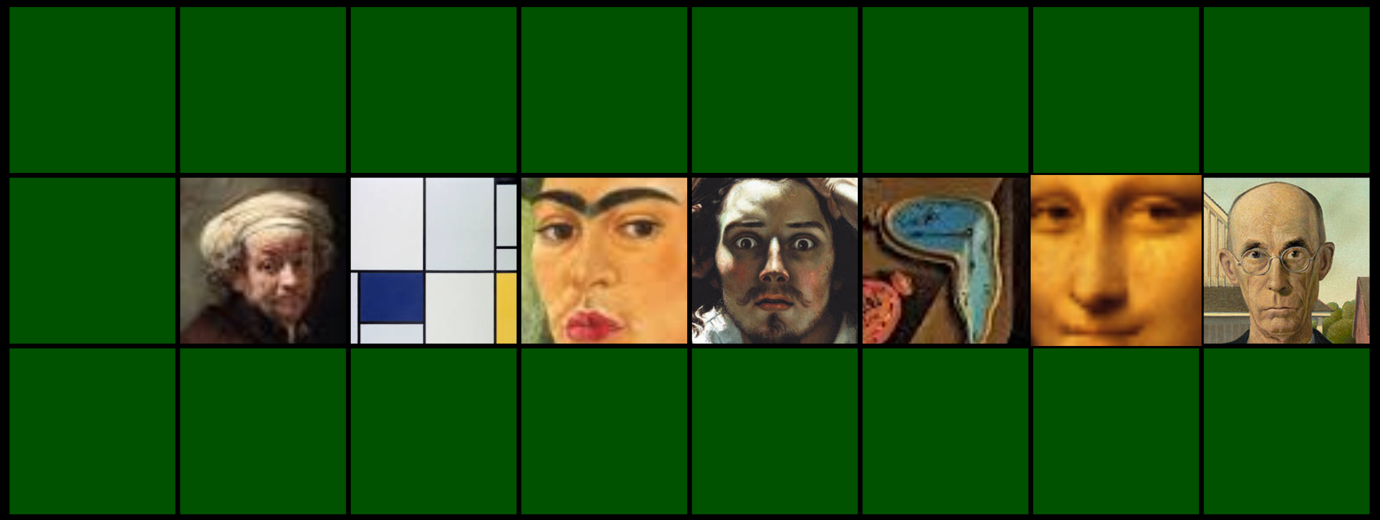
physical memory



attacker memory

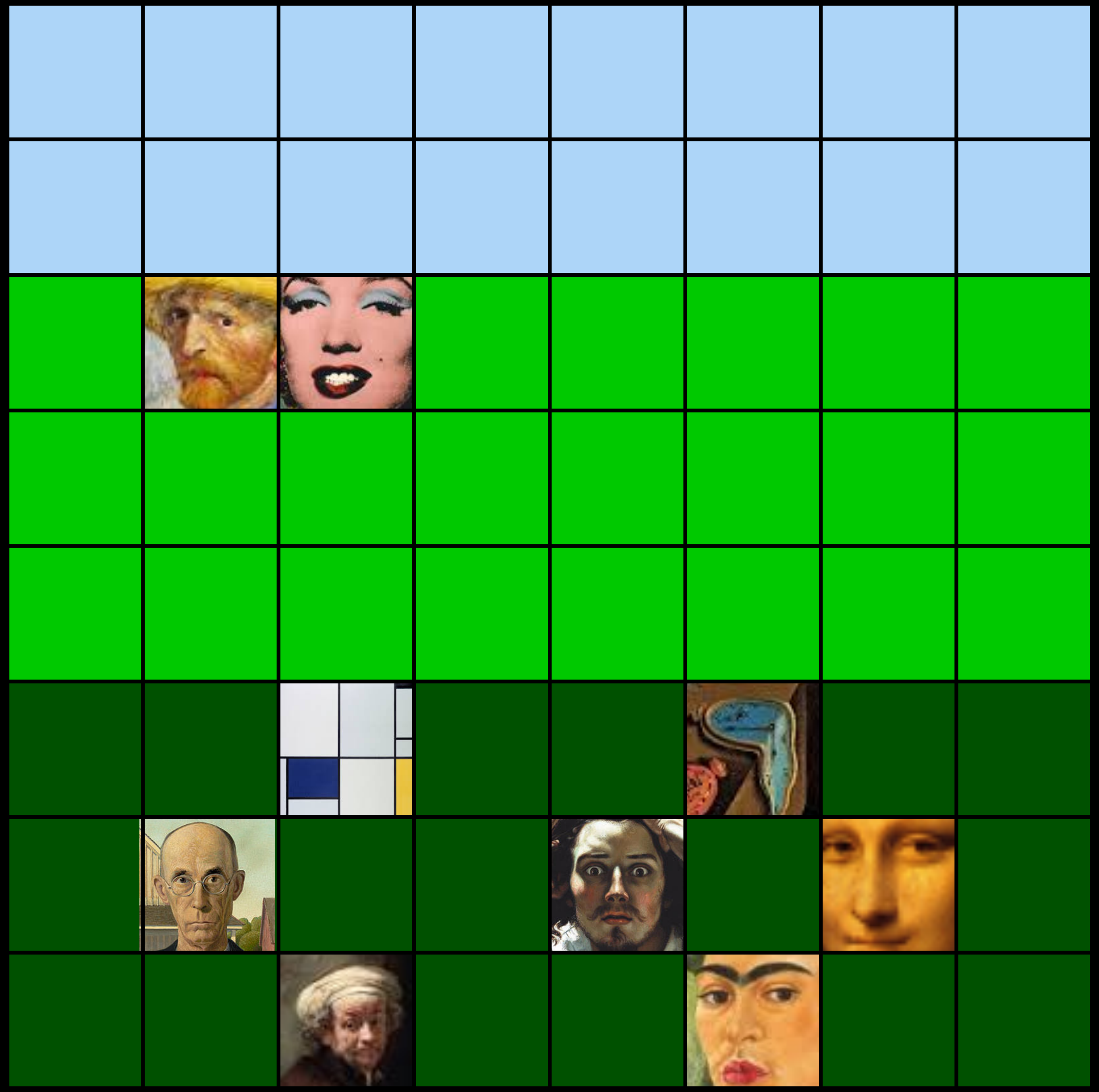


victim memory

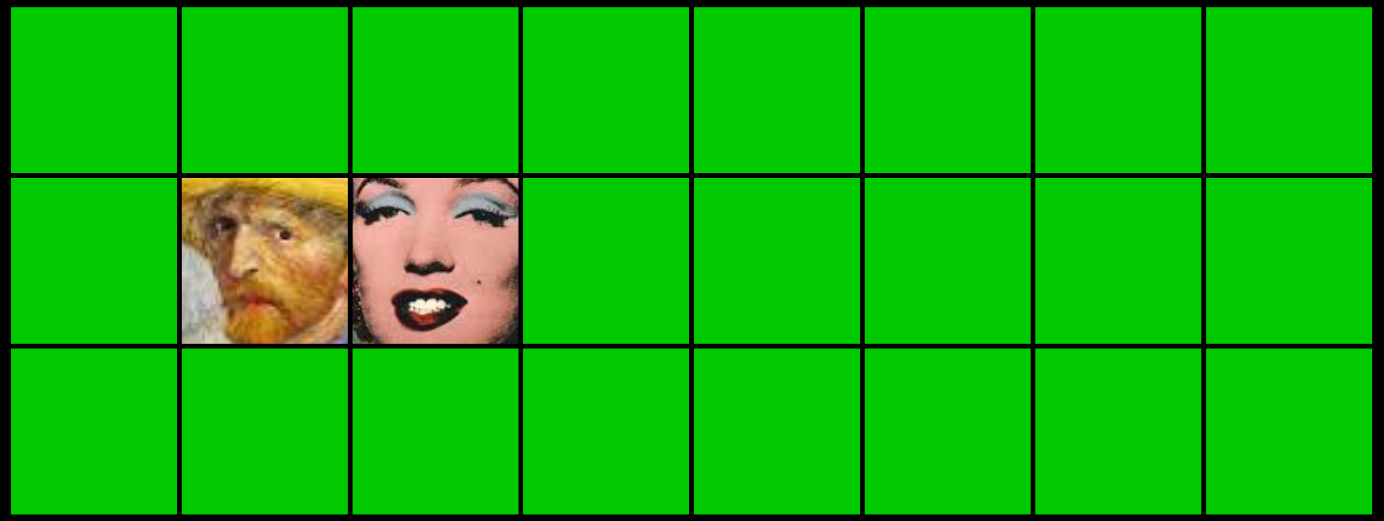


Primitive #3: birthday heapspray

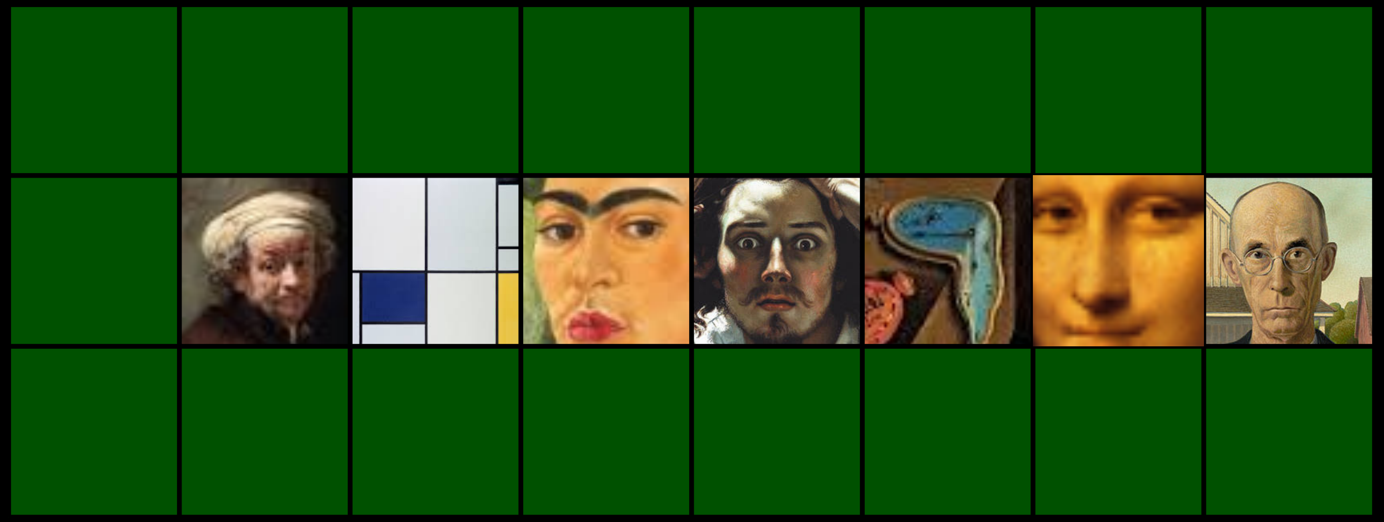
physical memory



attacker memory

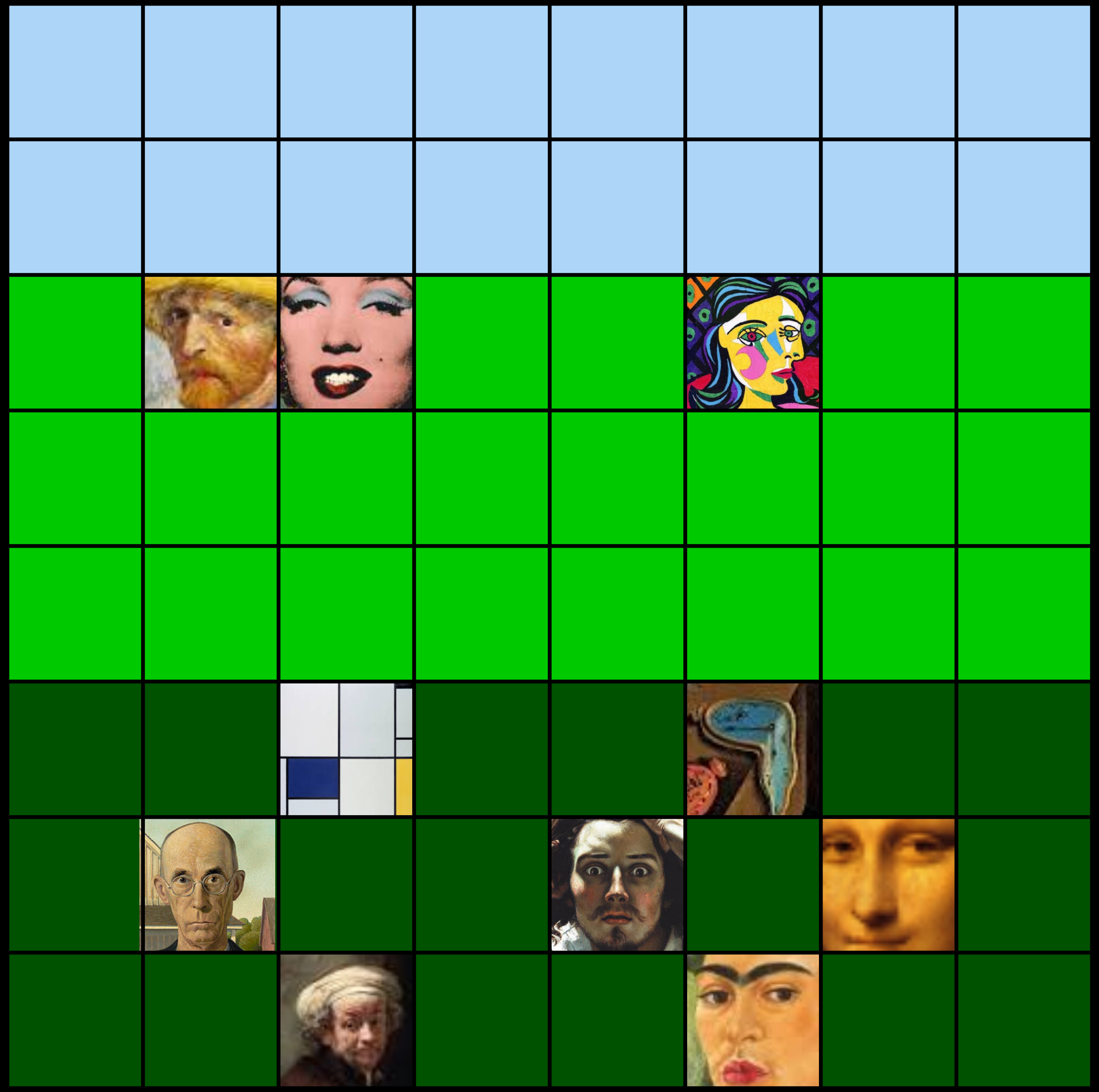


victim memory

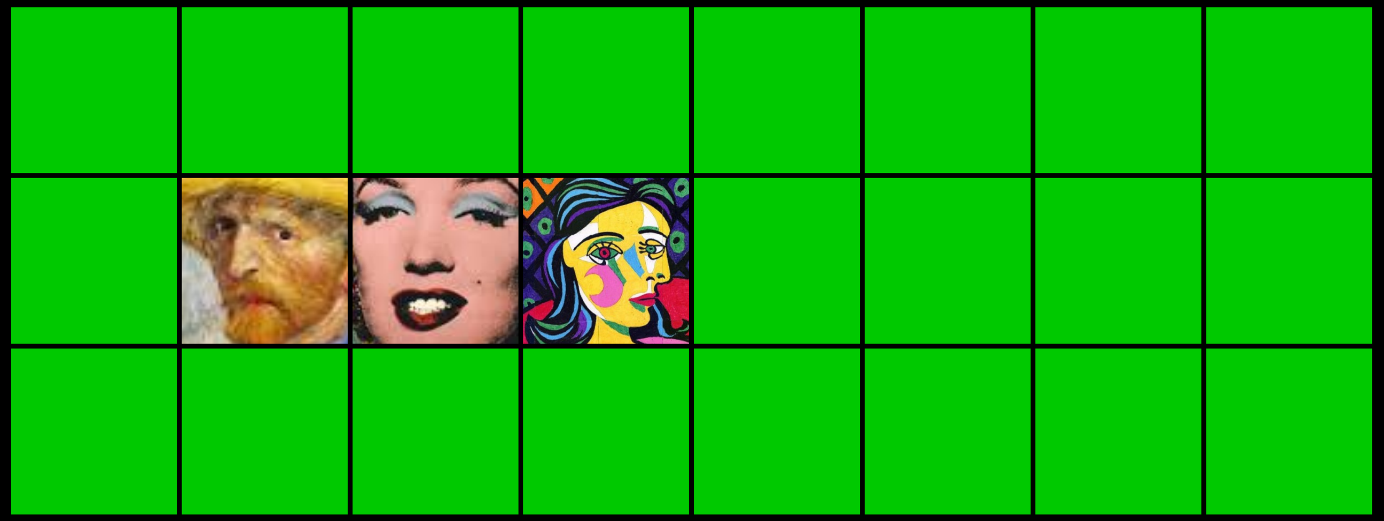


Primitive #3: birthday heapspray

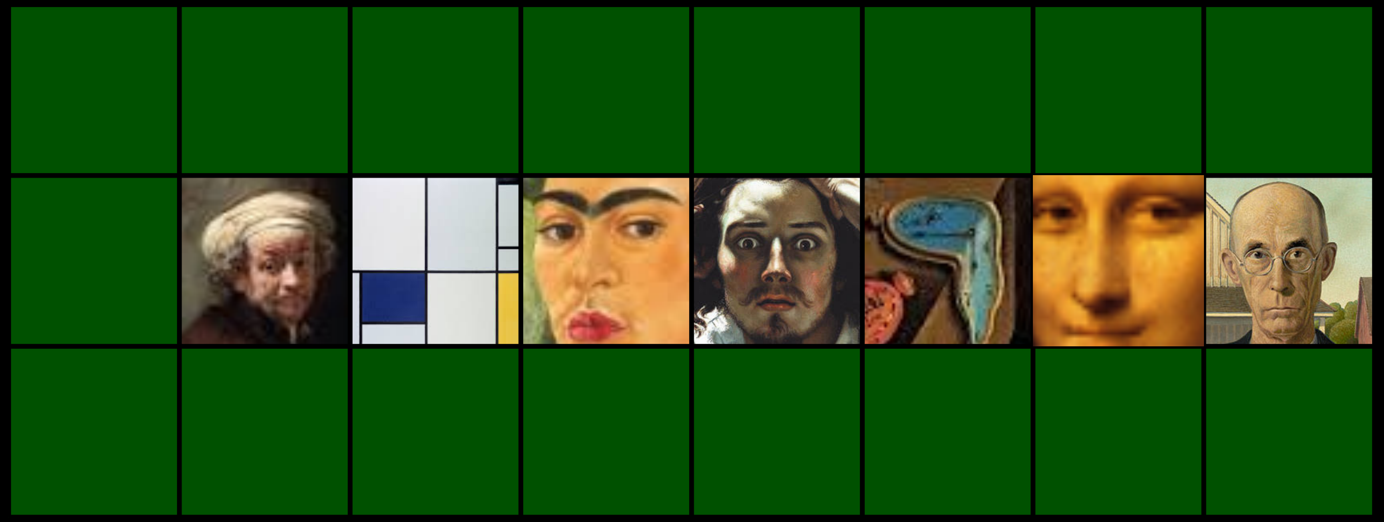
physical memory



attacker memory

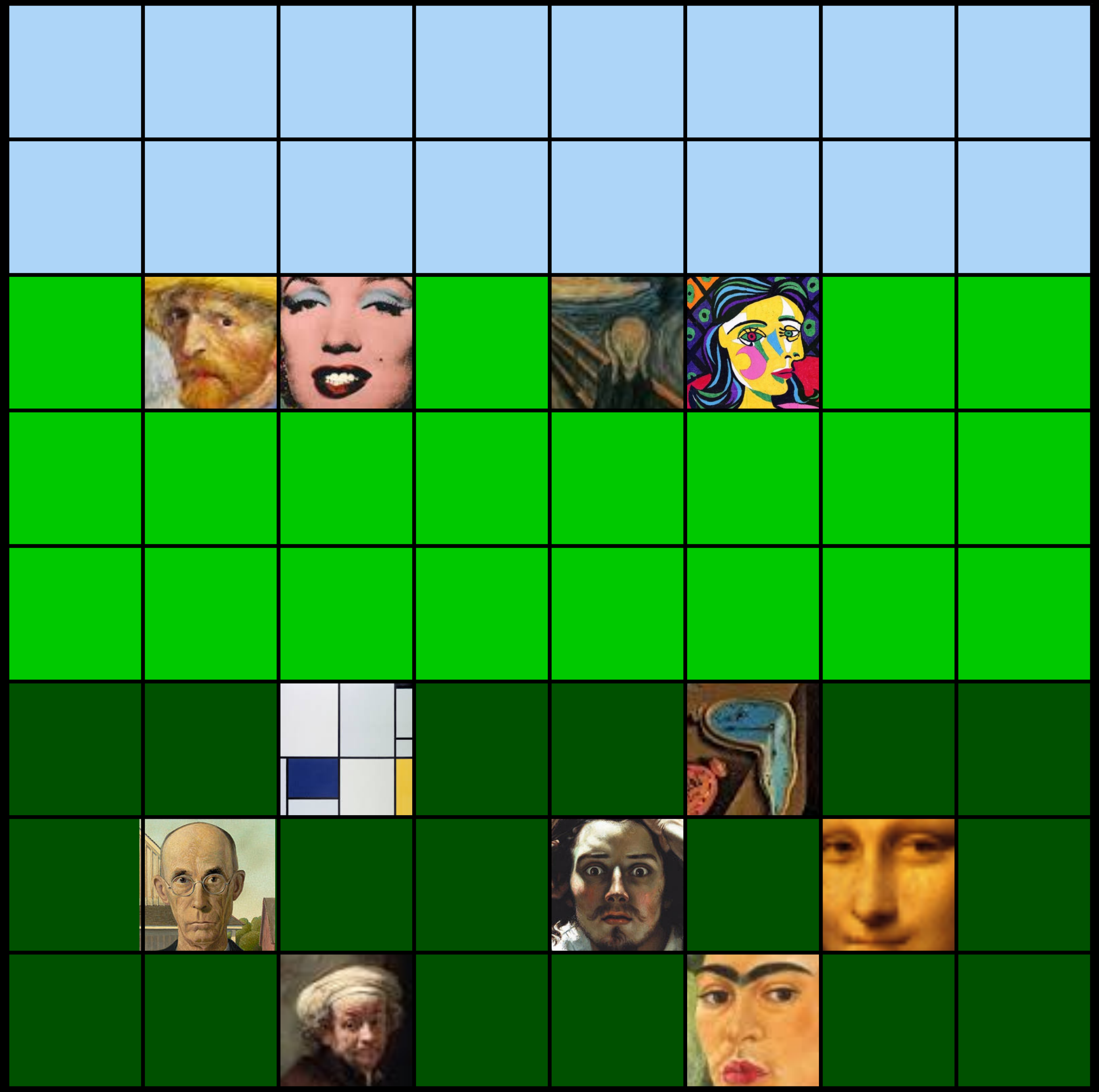


victim memory

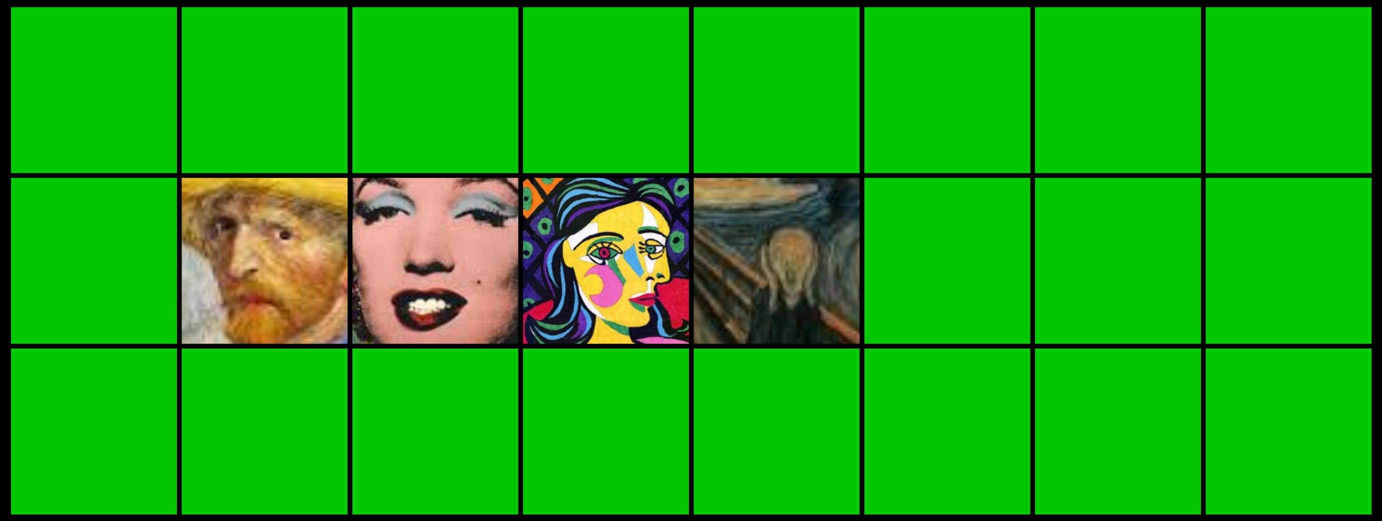


Primitive #3: birthday heapspray

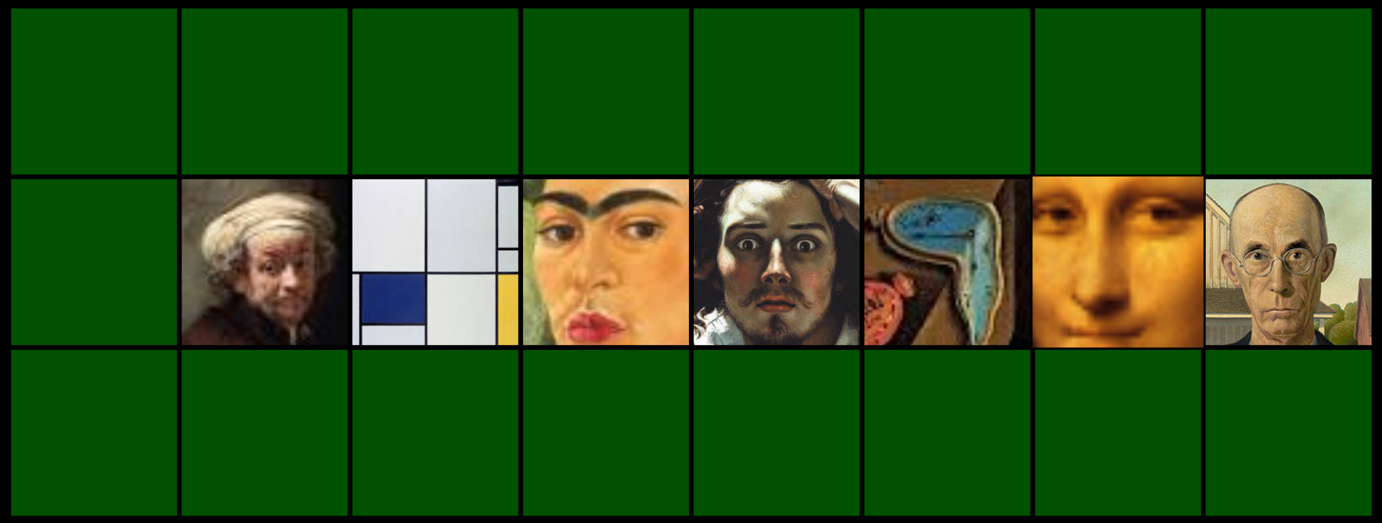
physical memory



attacker memory

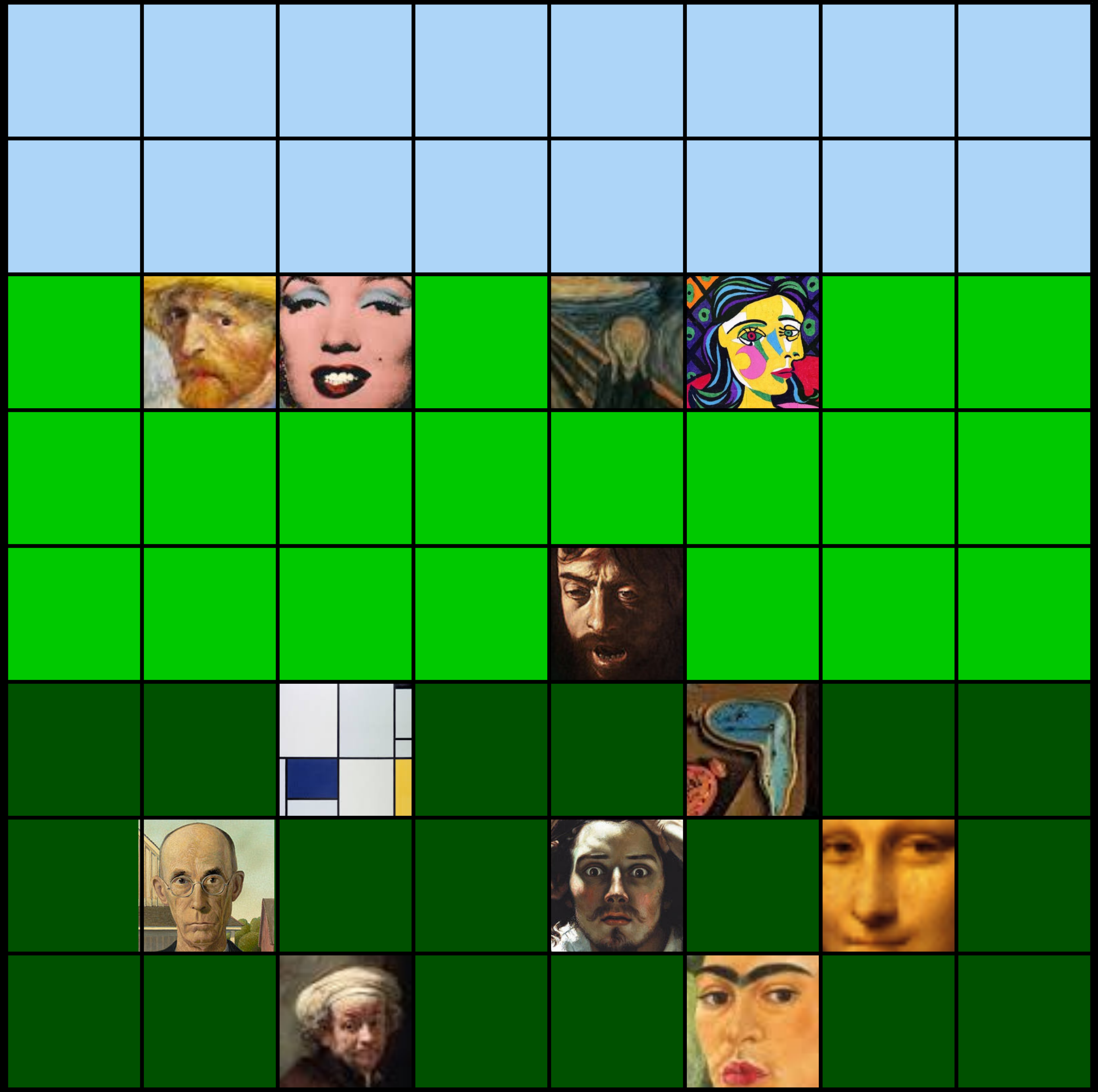


victim memory

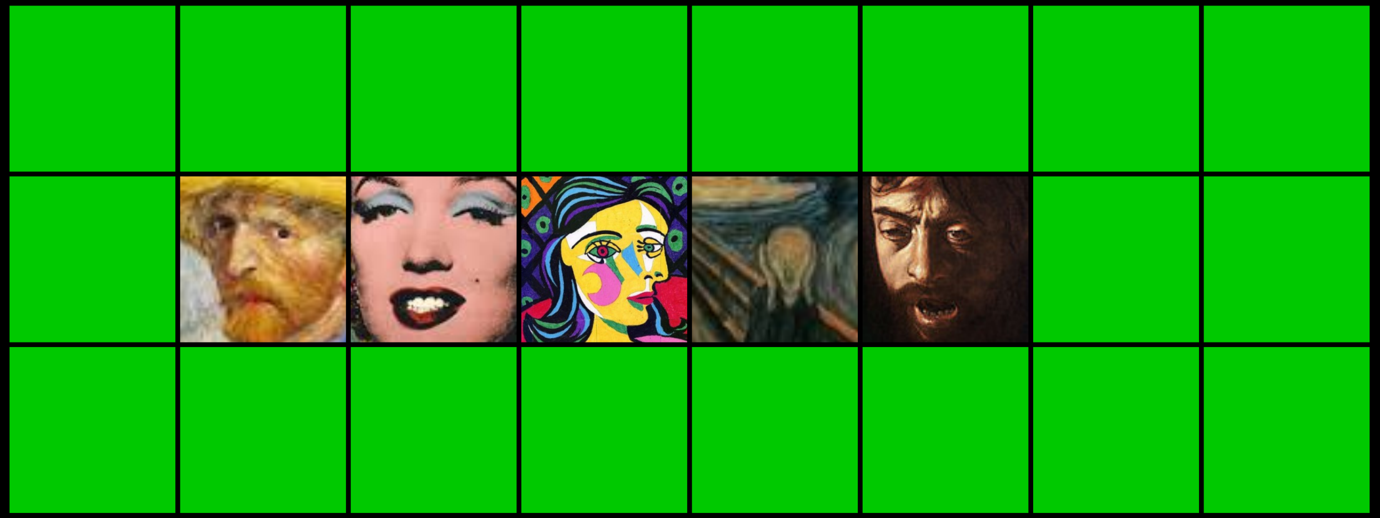


Primitive #3: birthday heapspray

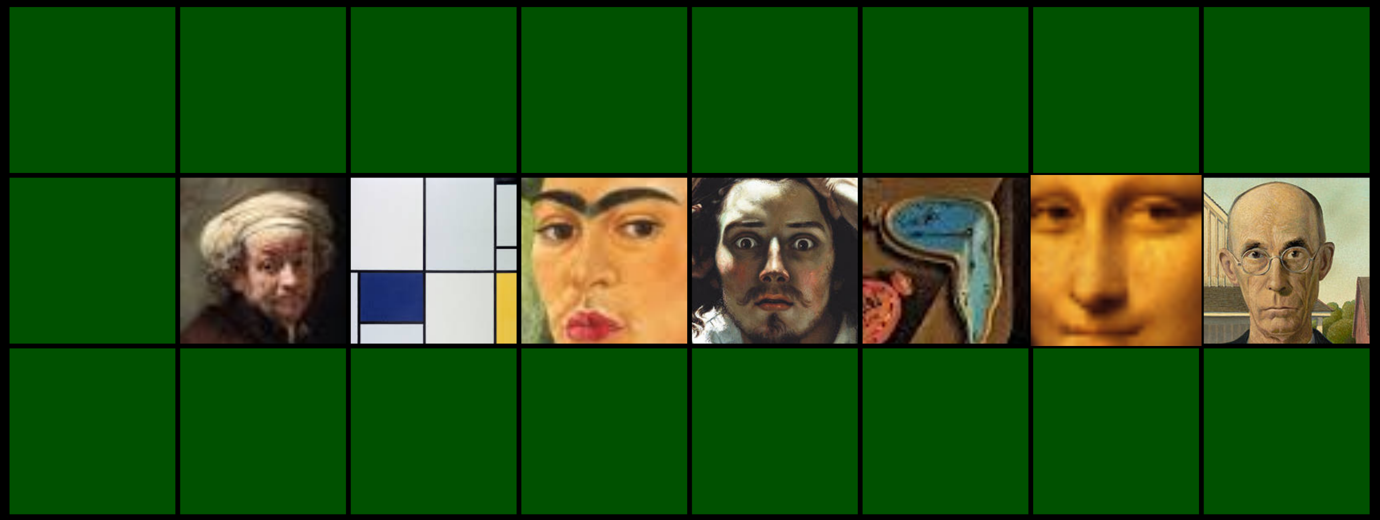
physical memory



attacker memory

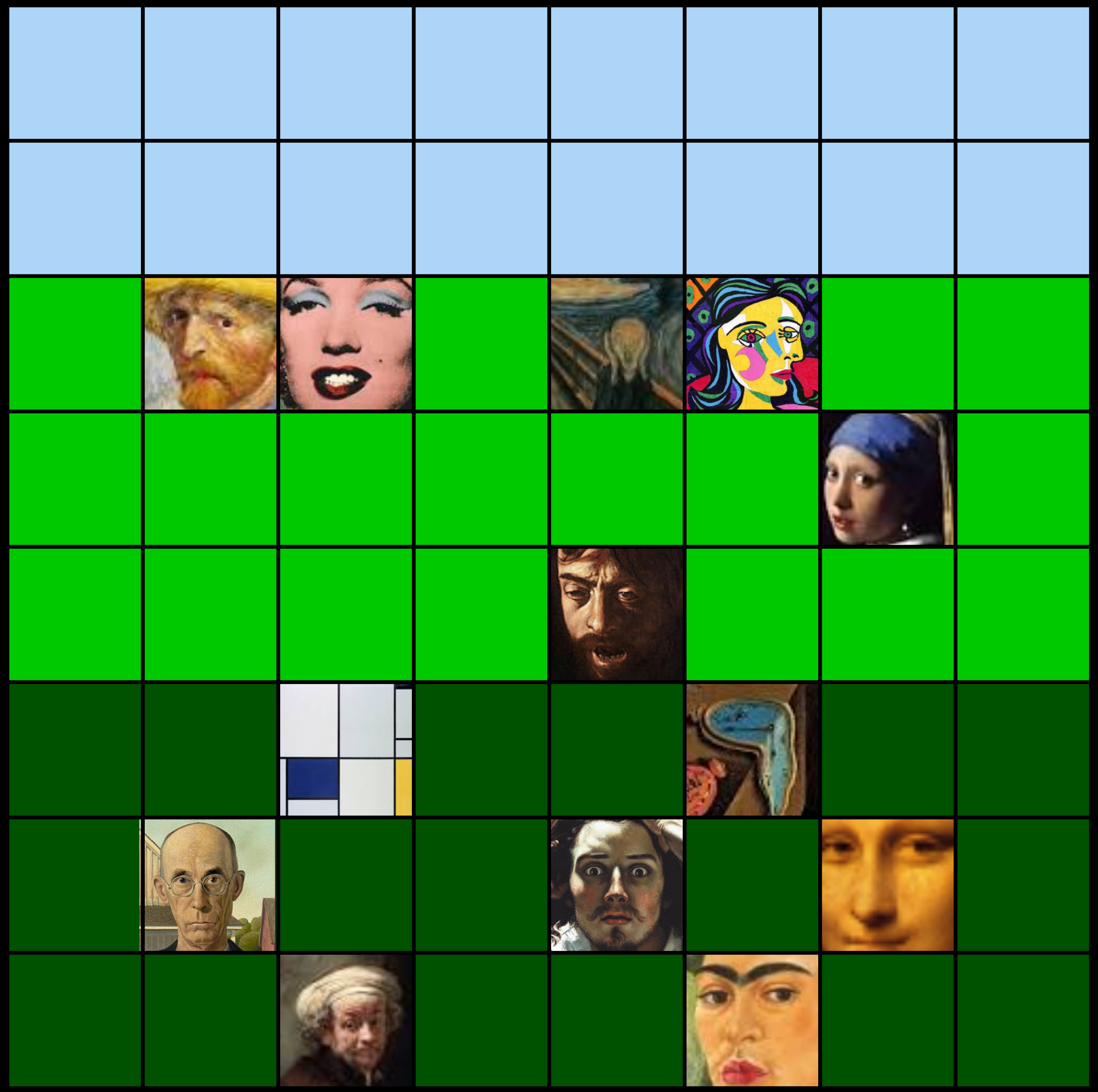


victim memory

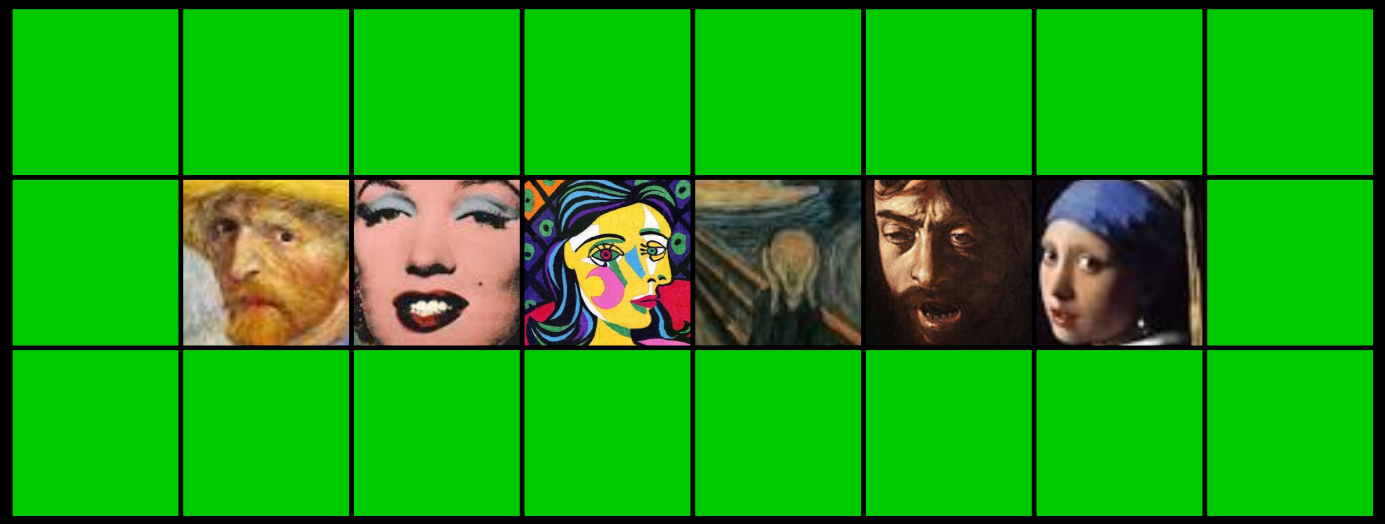


Primitive #3: birthday heapspray

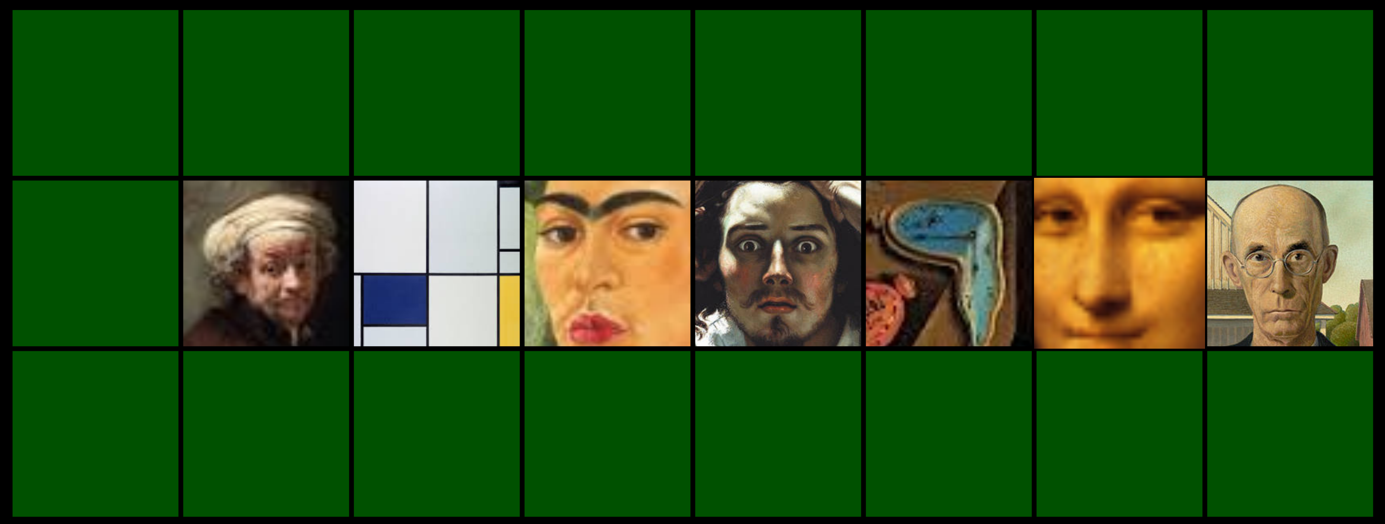
physical memory



attacker memory

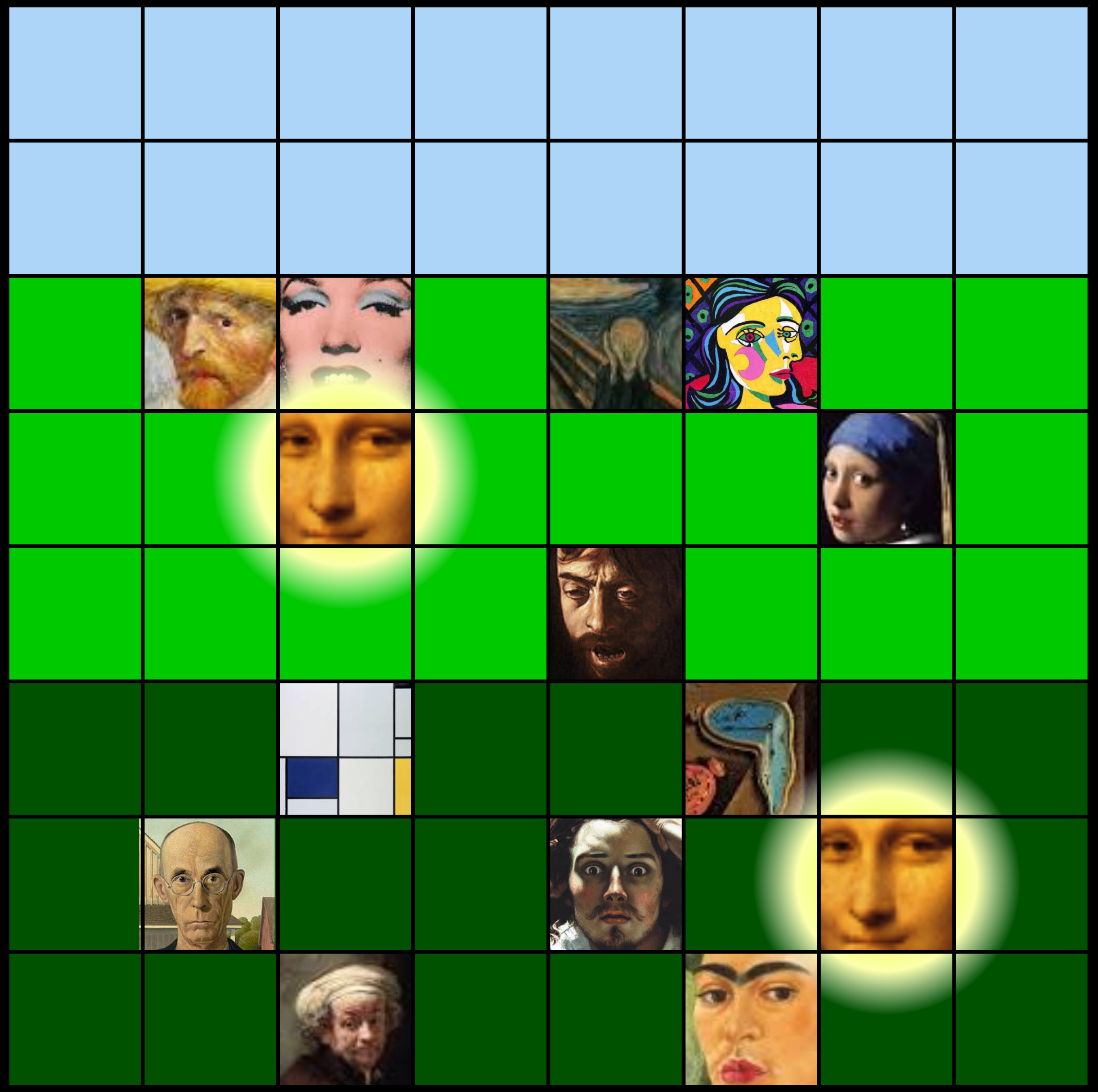


victim memory

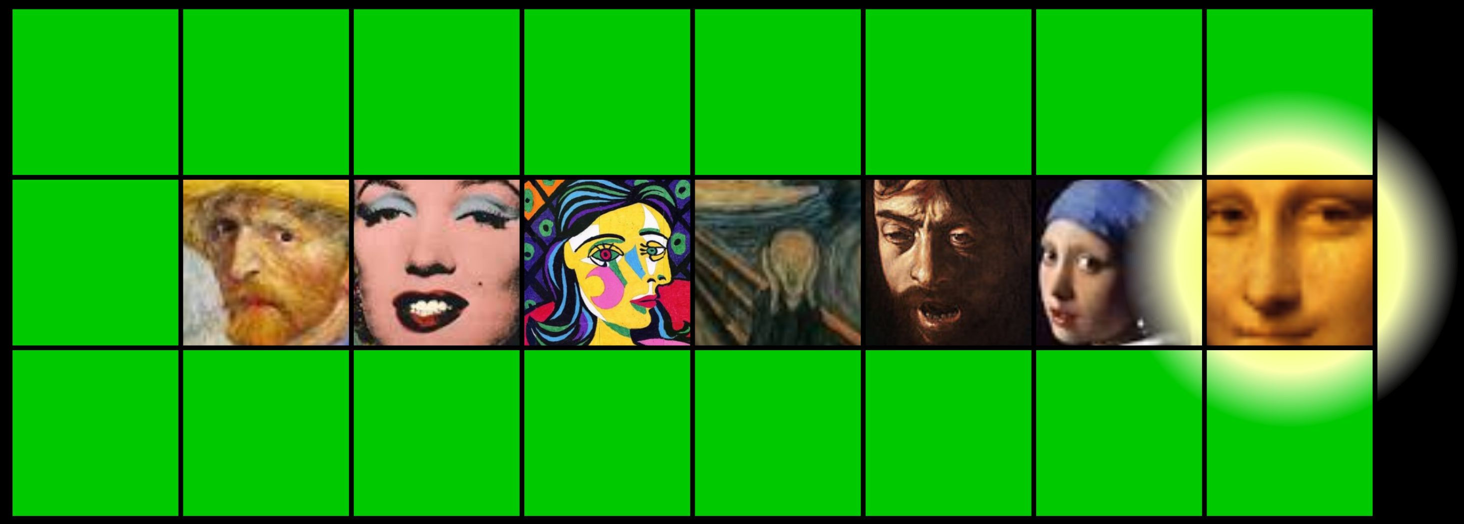


Primitive #3: birthday heapspray

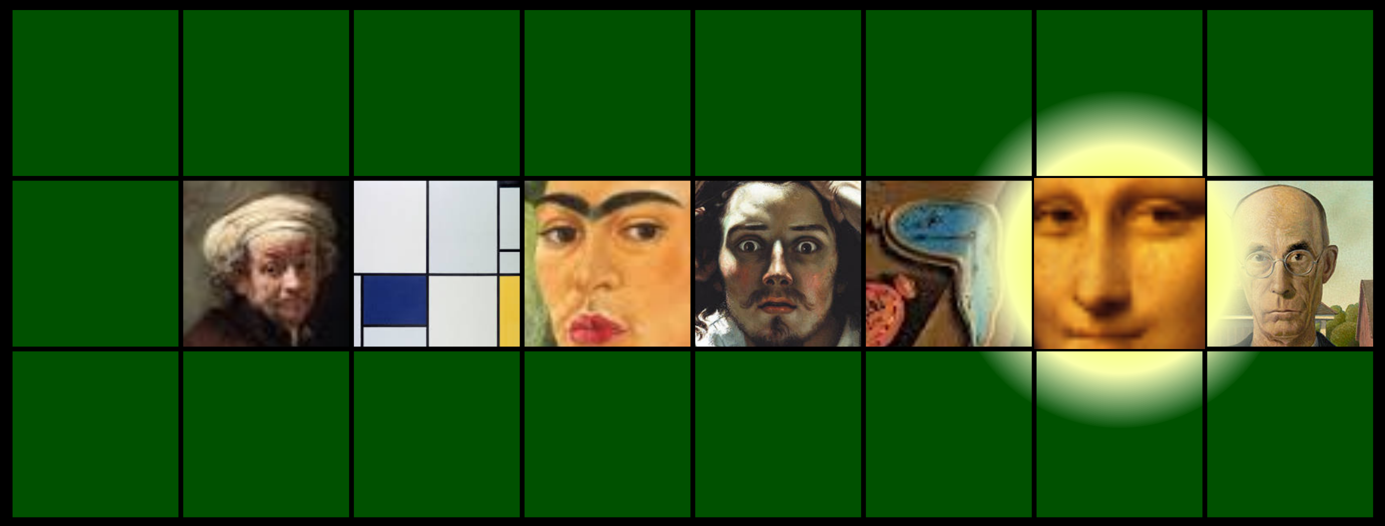
physical memory



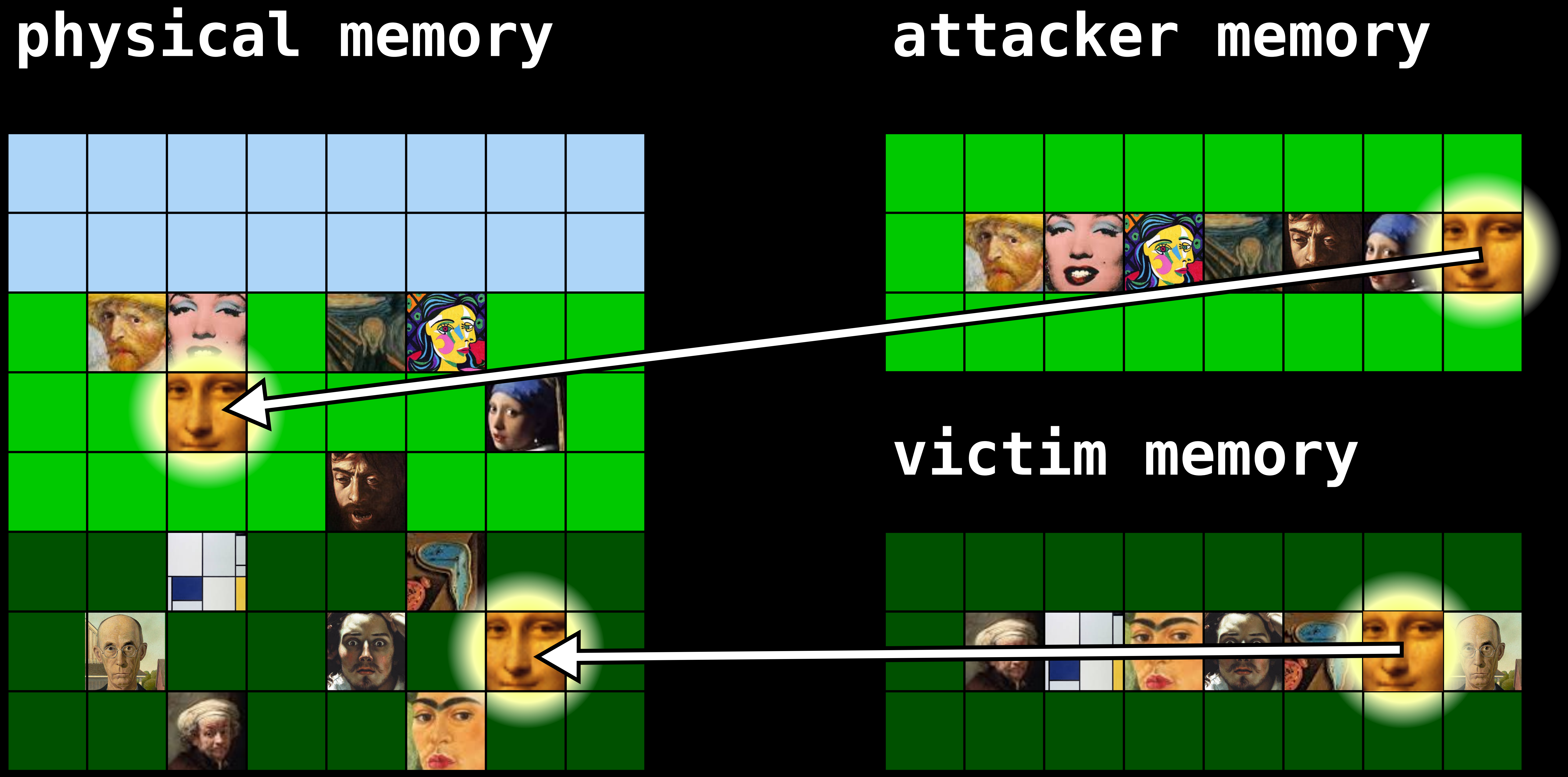
attacker memory



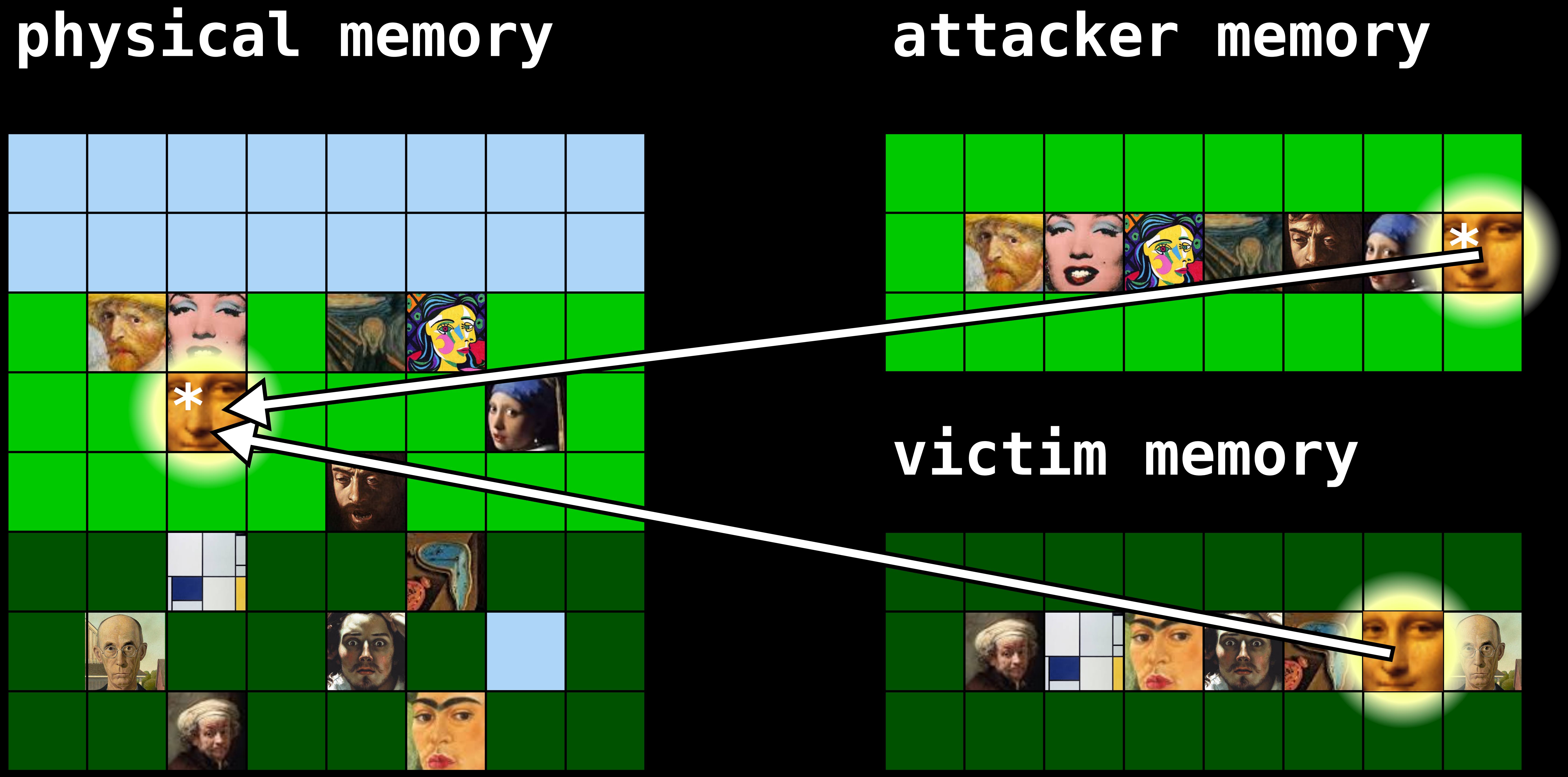
victim memory



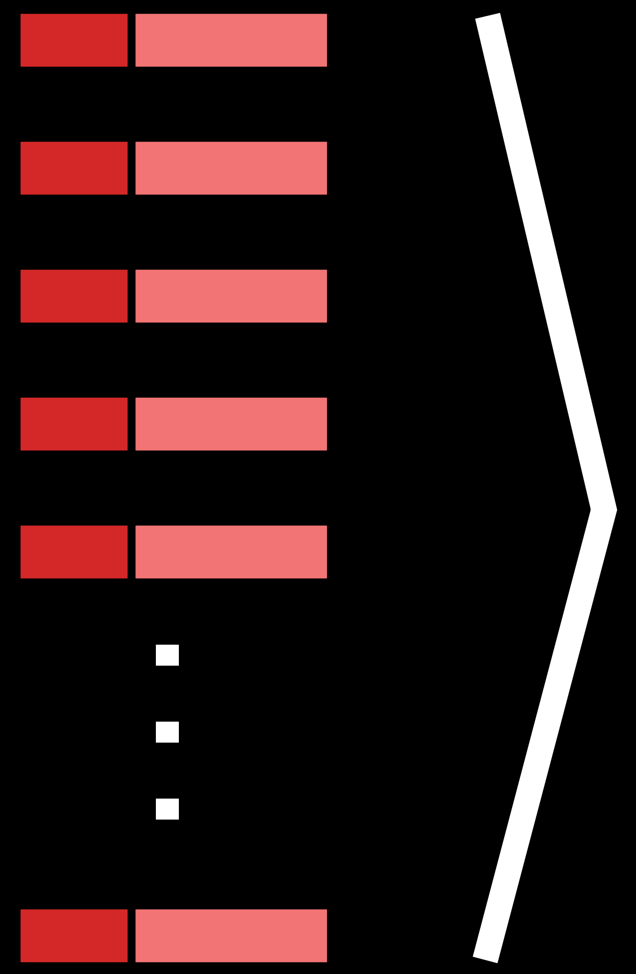
Primitive #3: birthday heapspray



Primitive #3: birthday heapspray

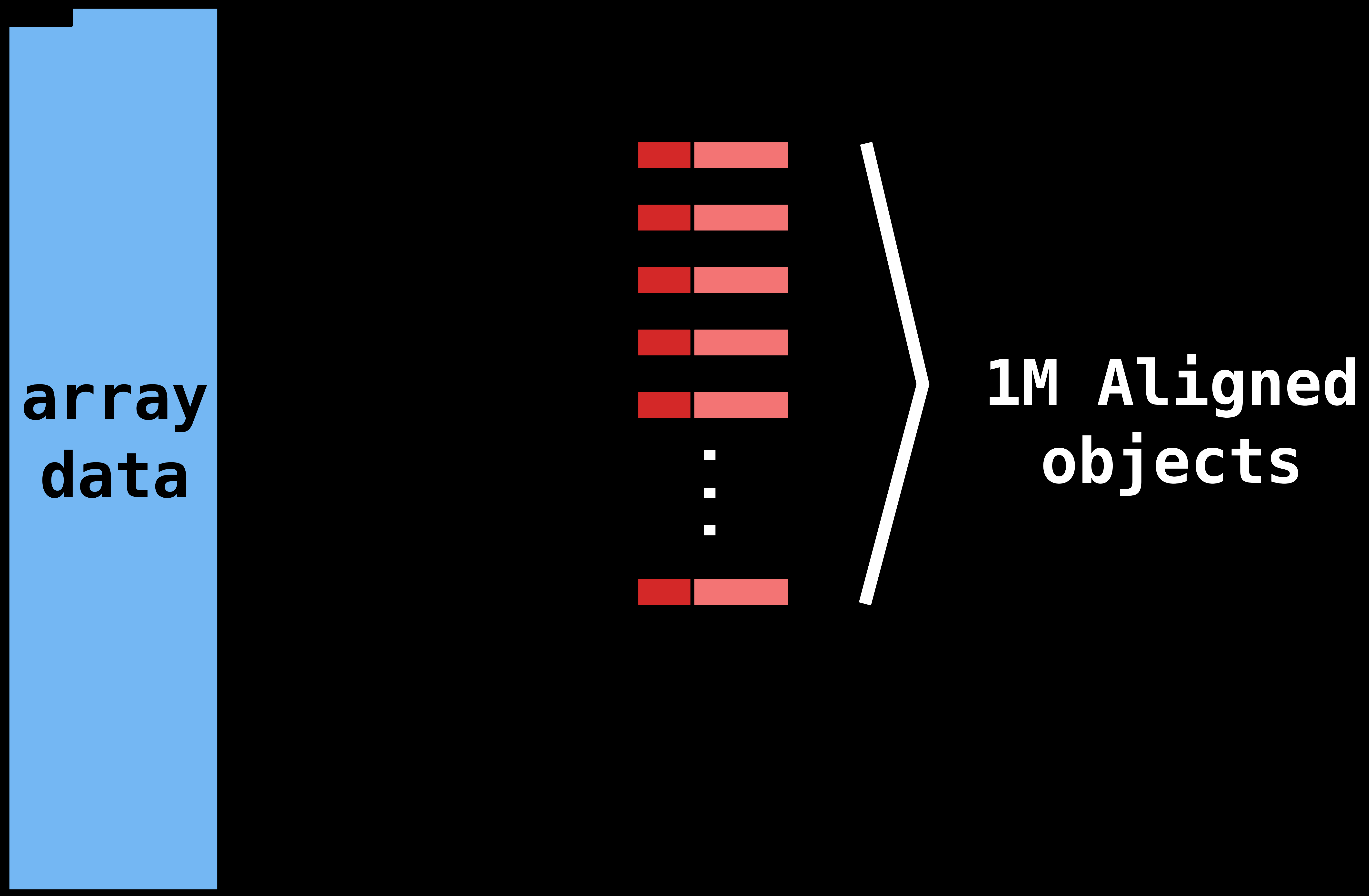


Creating Secret Pages

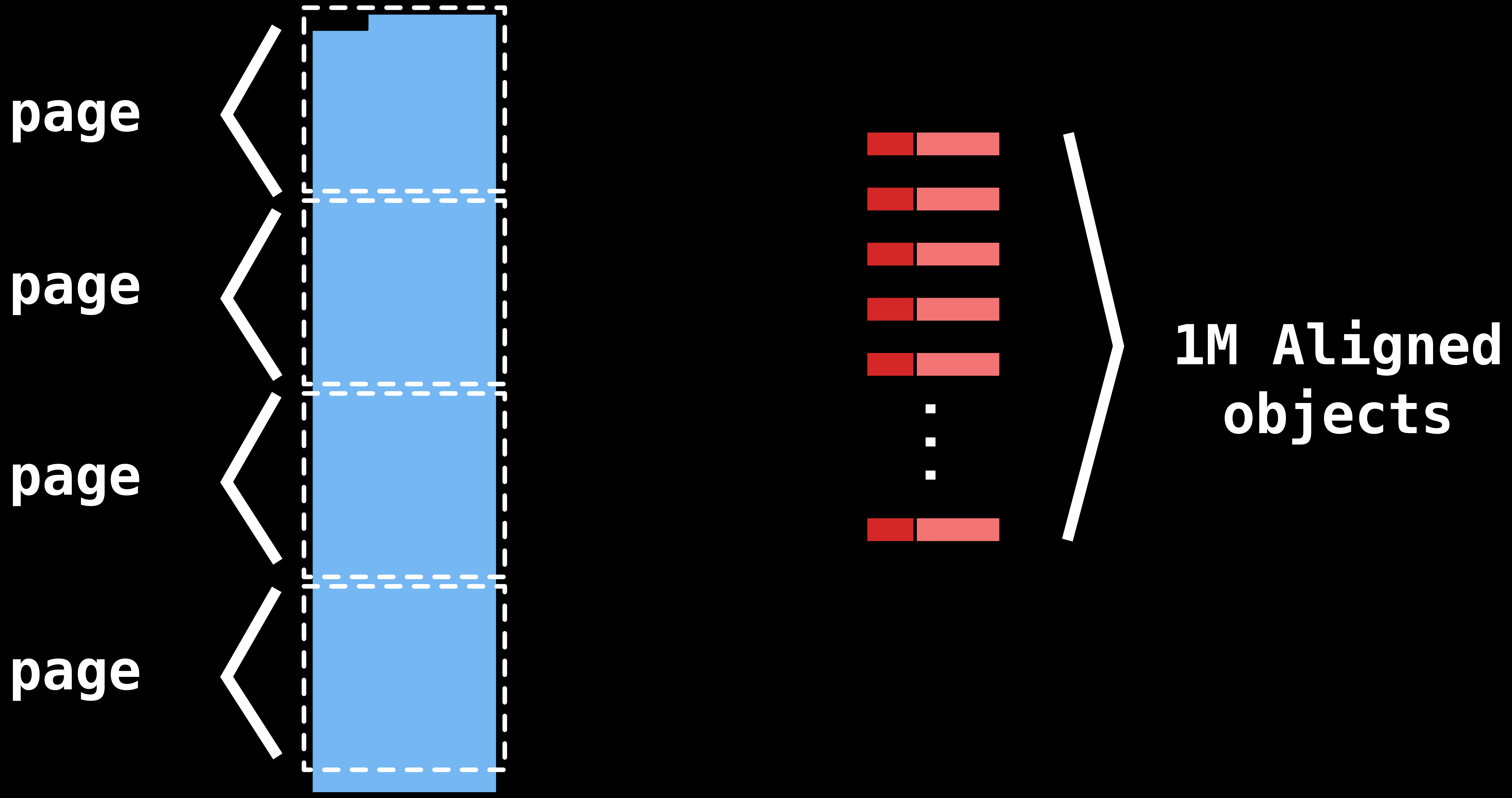


1M Aligned objects

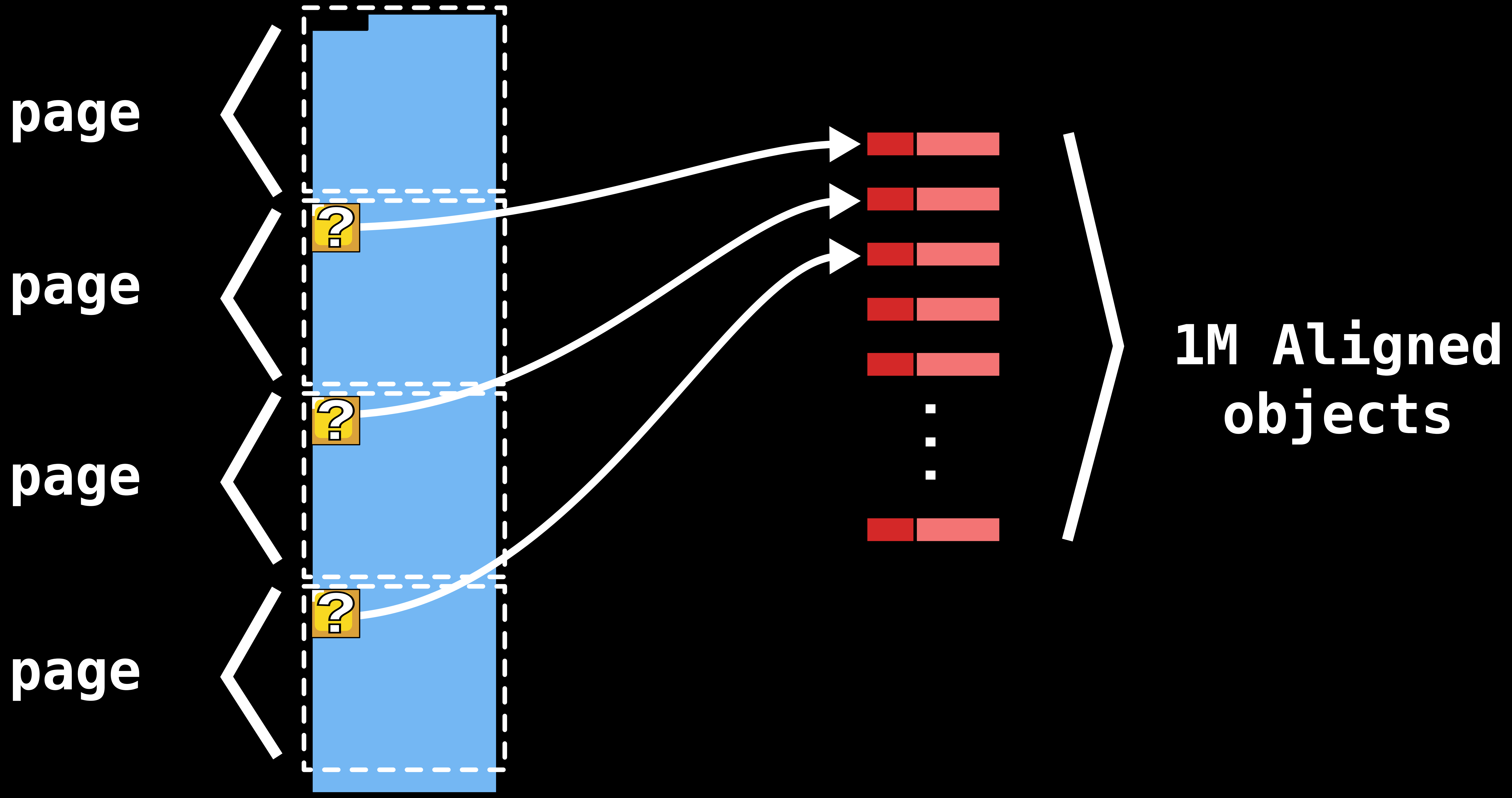
Creating Secret Pages



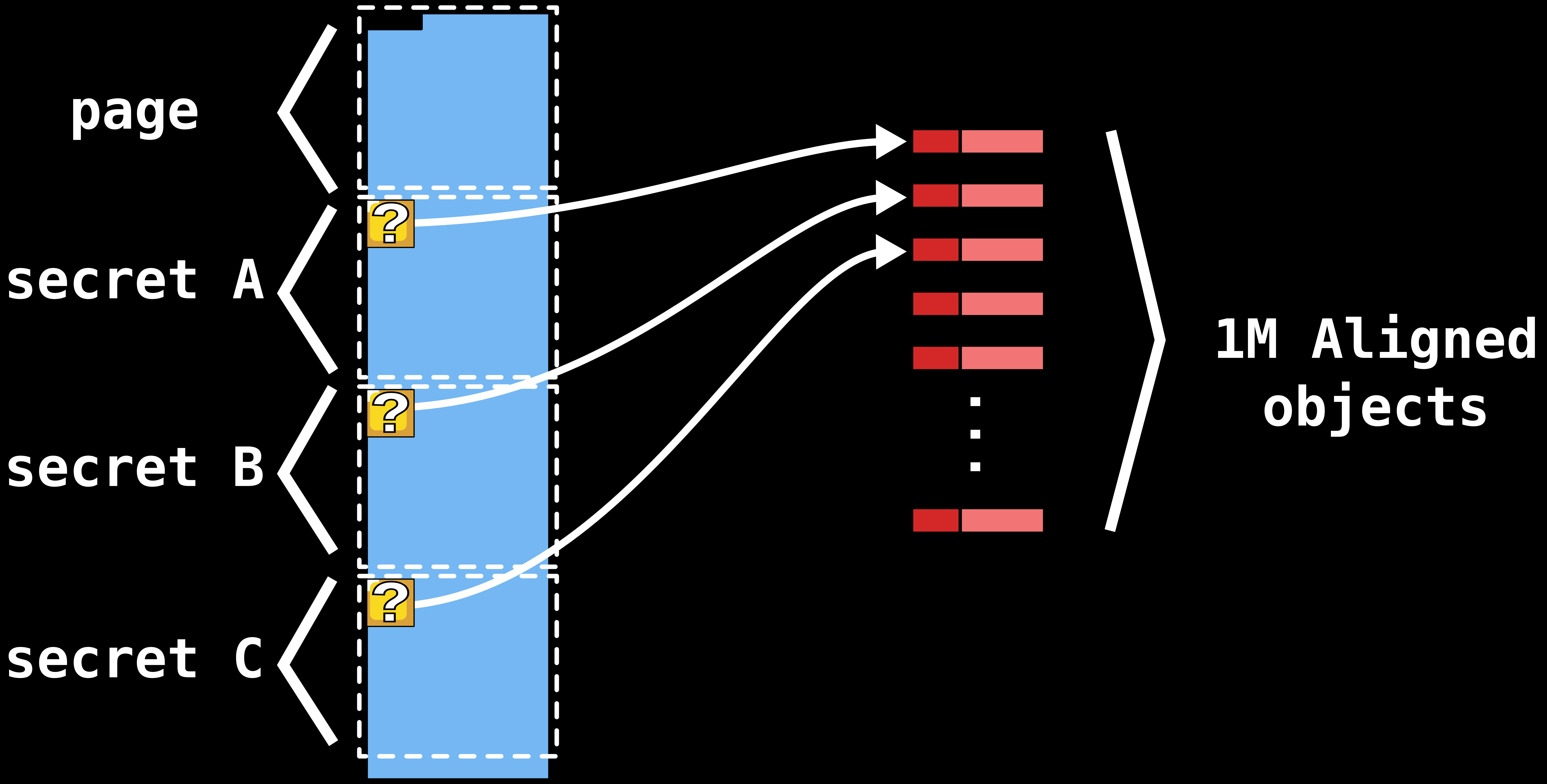
Creating Secret Pages



Creating Secret Pages



Creating Secret Pages

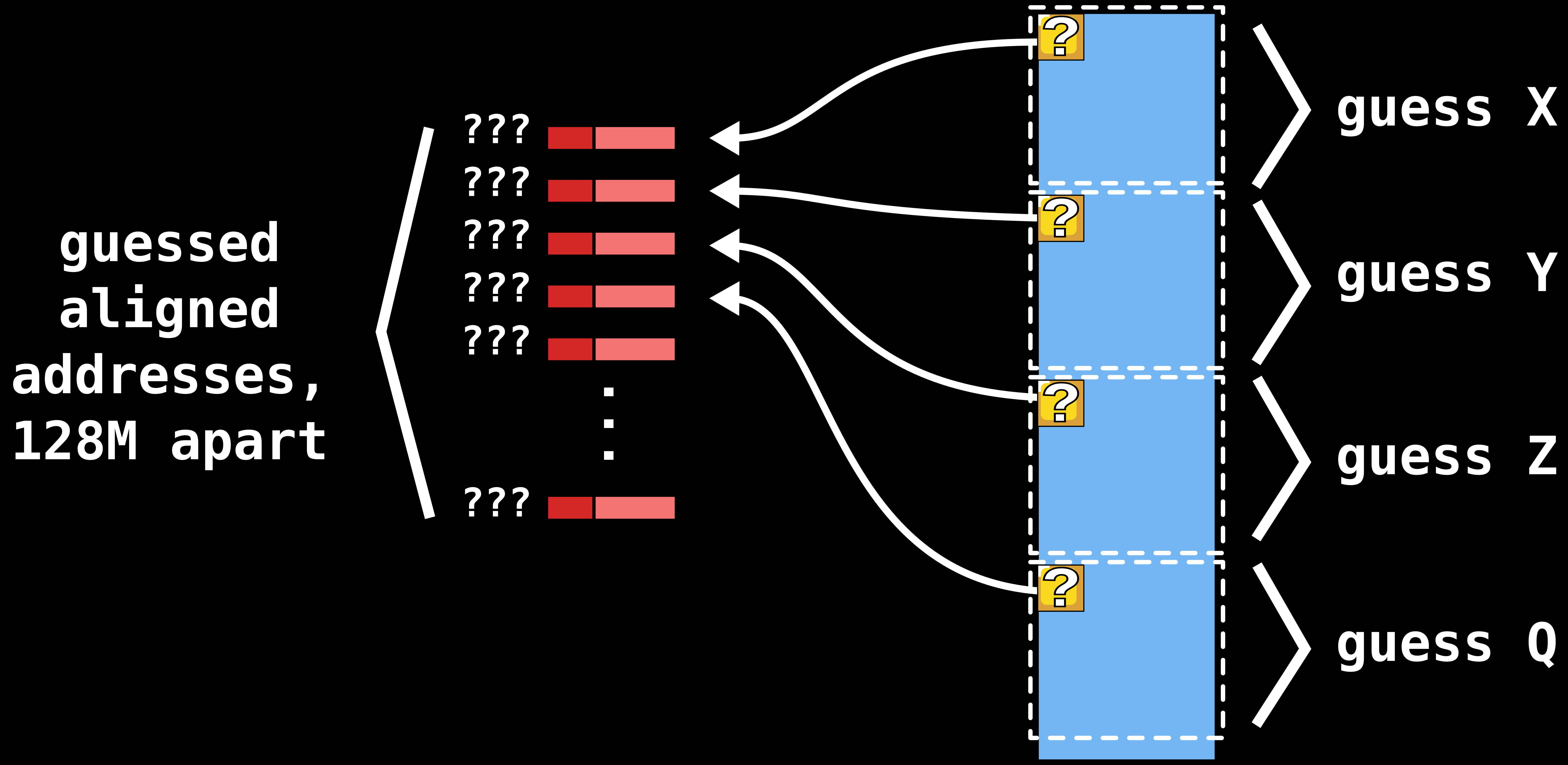


Creating Guess Pages

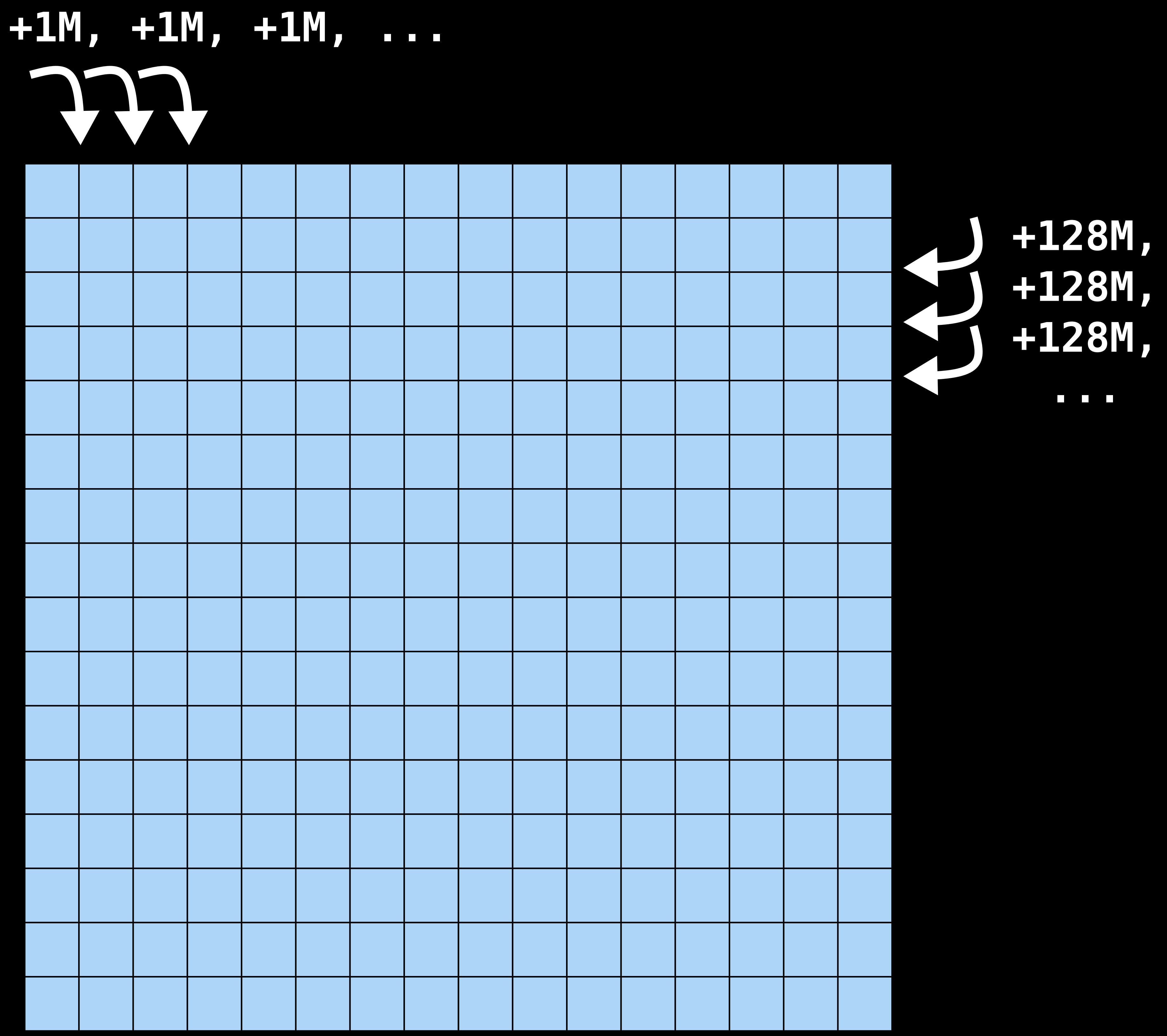
typed
array
data

Creating Guess Pages

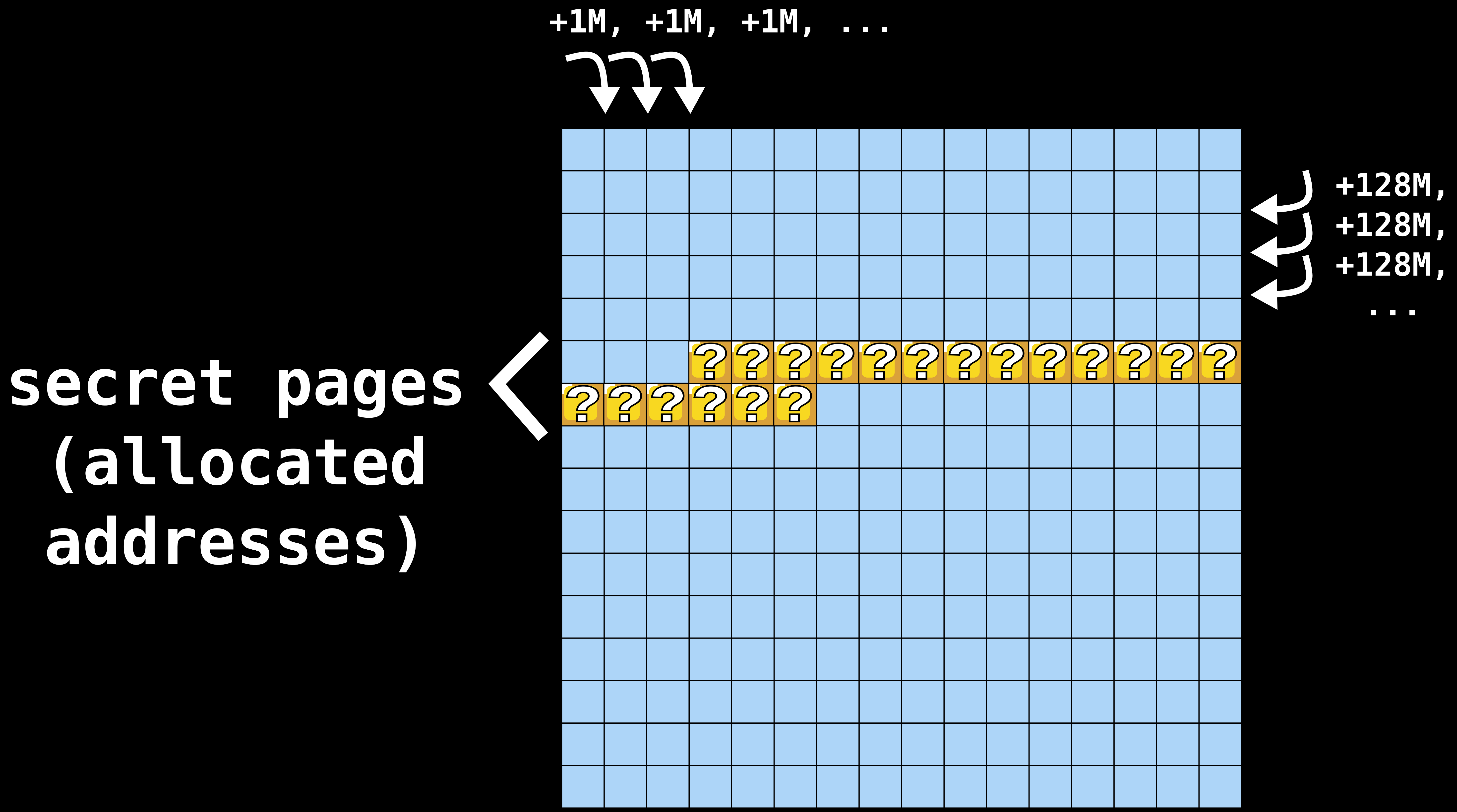
Creating Guess Pages



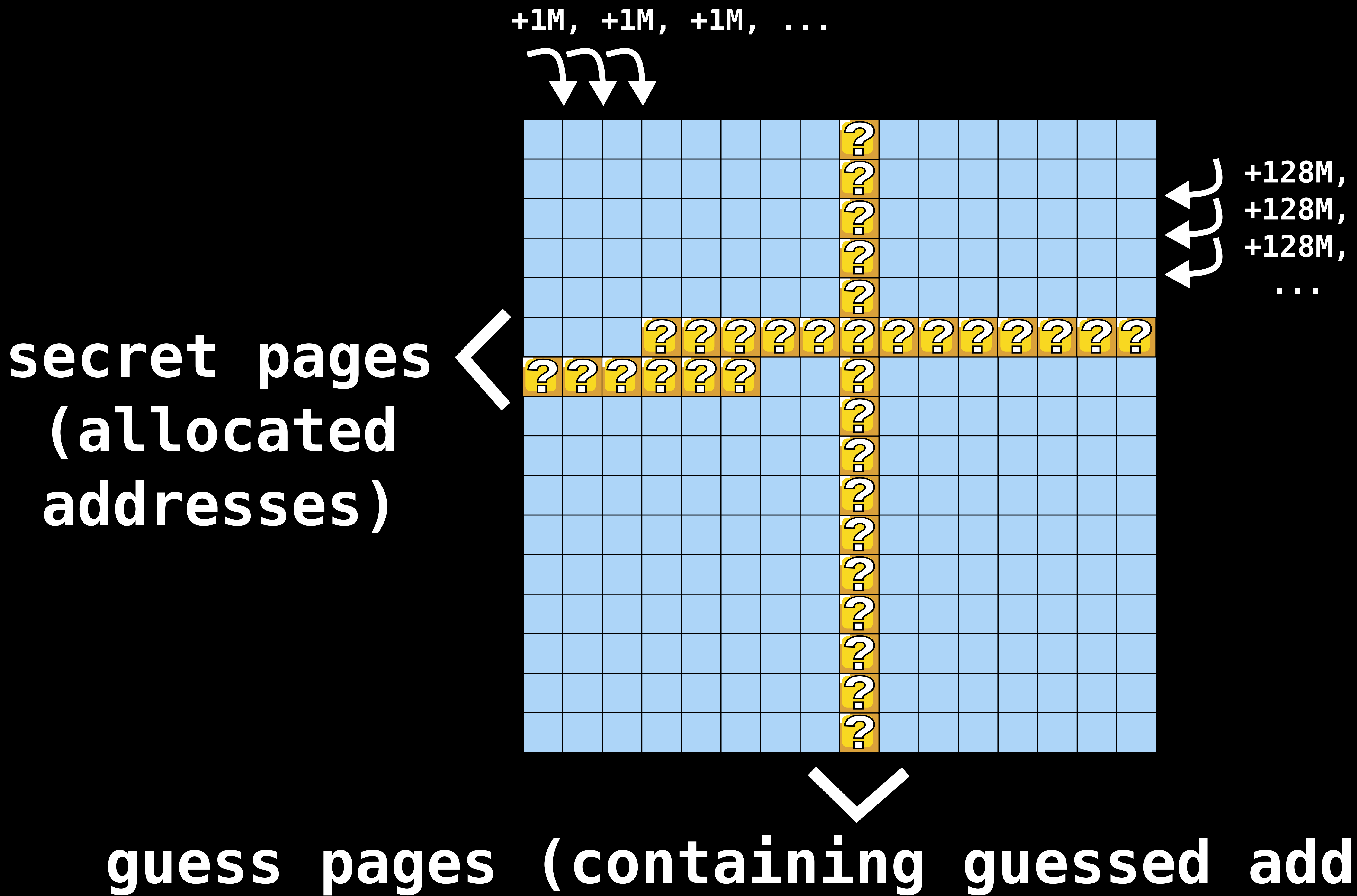
Birthday heap spray



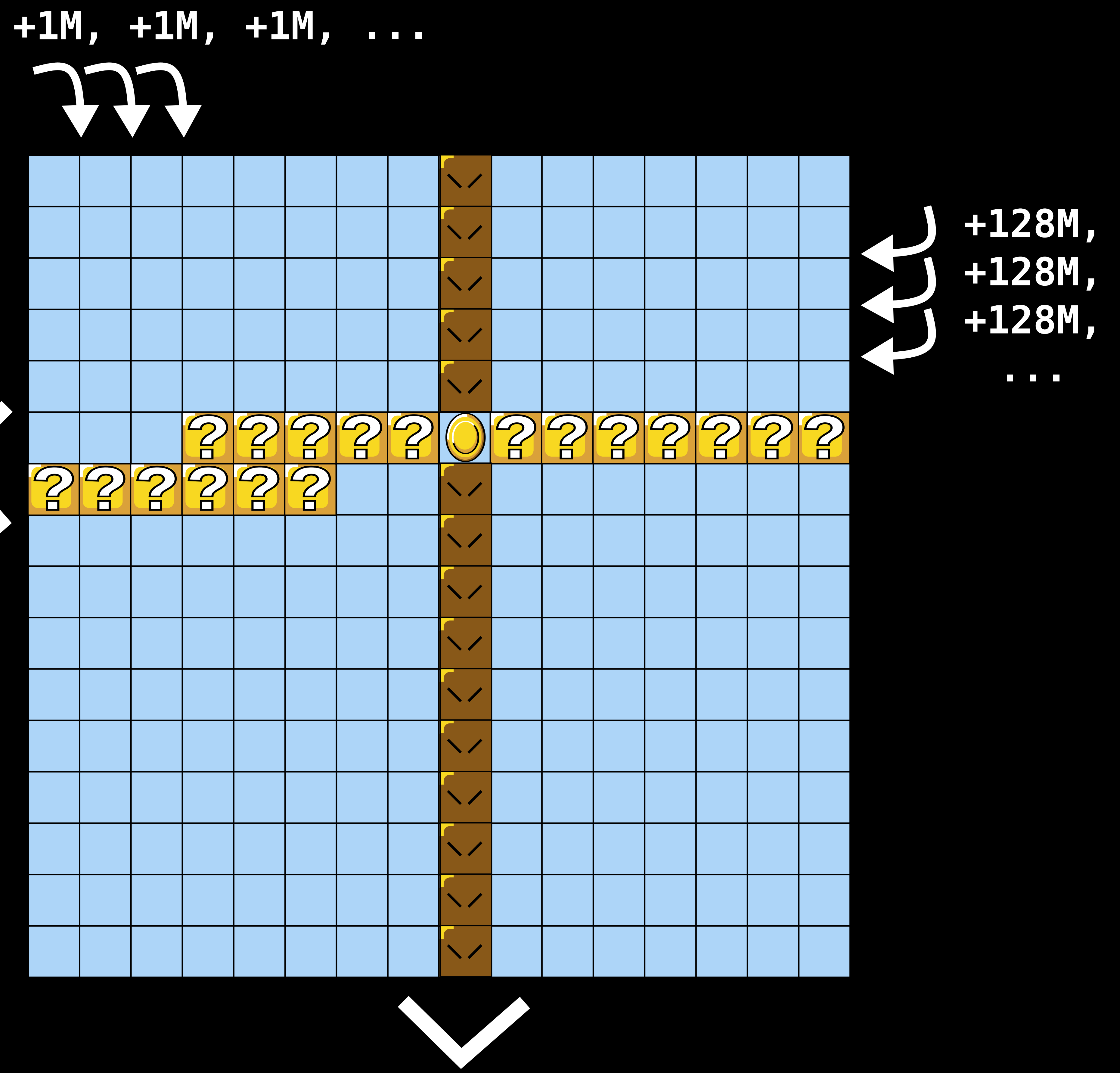
Birthday heap spray



Birthday heap spray



Birthday heap spray



secret pages (allocated addresses)

guess pages (containing guessed addresses)

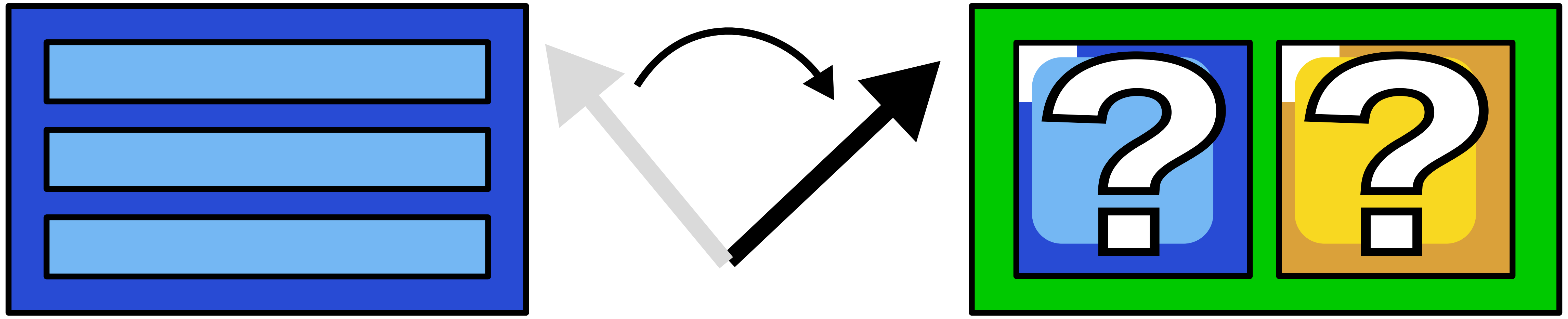
Outline:

Deduplication

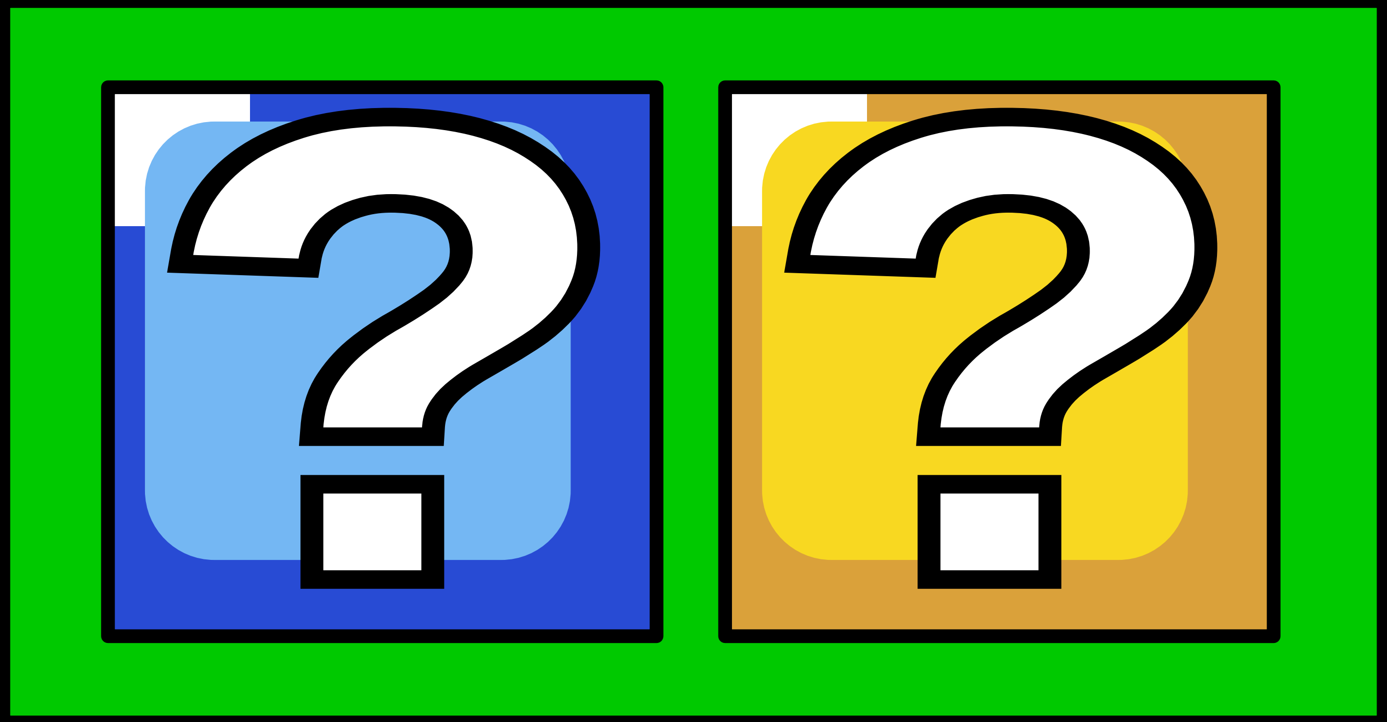
- leak heap & code addresses
- create a fake object

Rowhammer

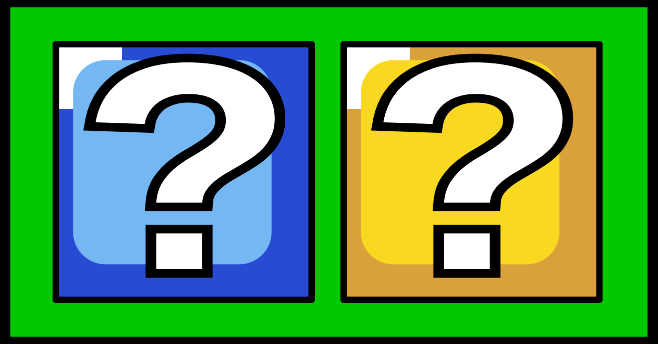
- create reference to our fake object



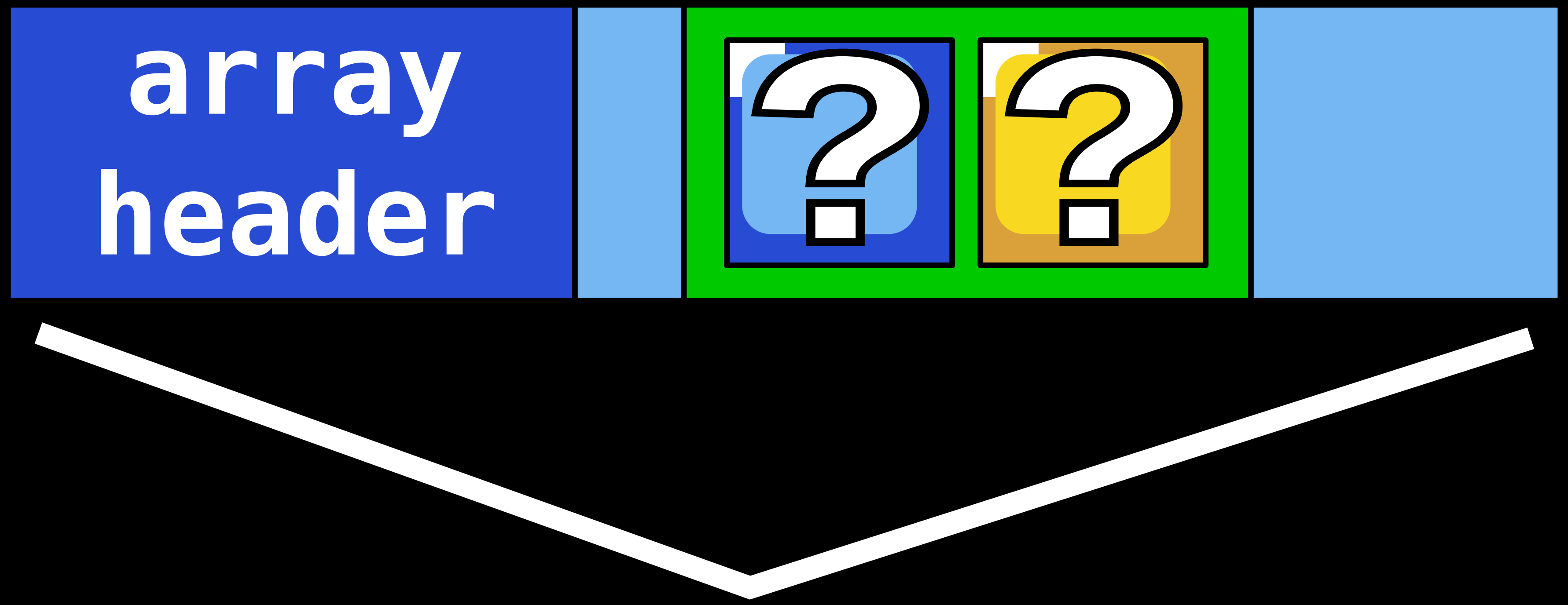
Fake Uint8Array object



Pointer pivotting

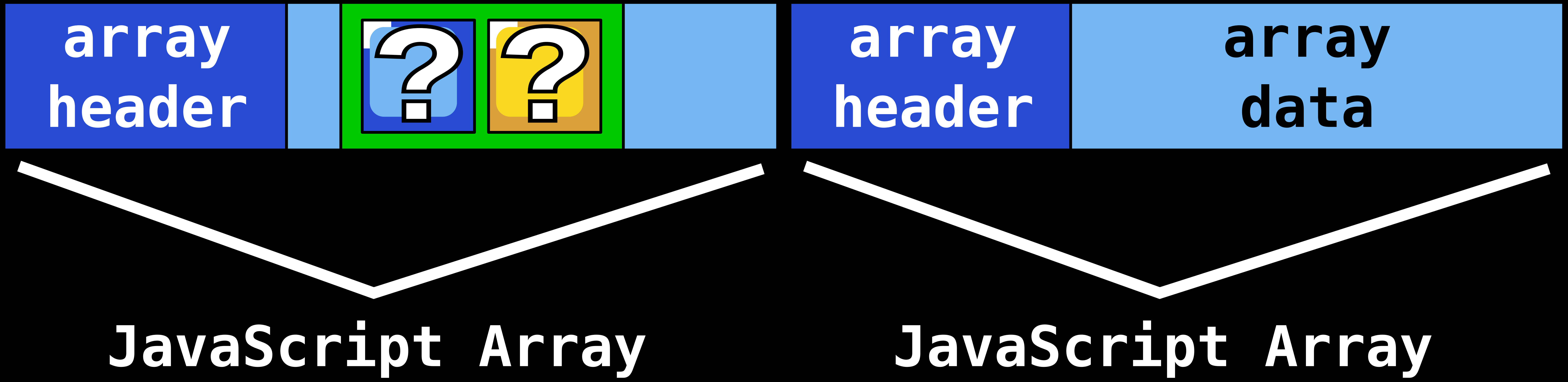


Pointer pivotting

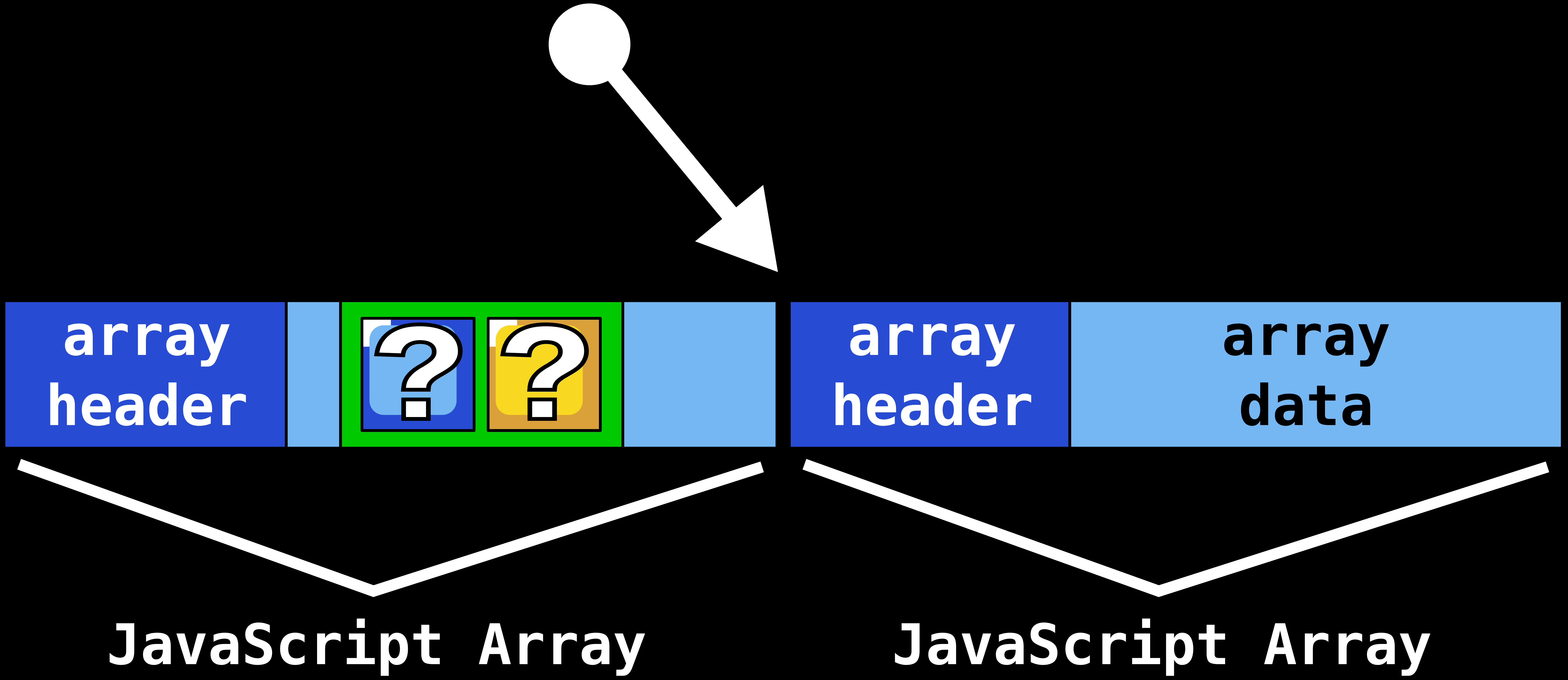


JavaScript Array

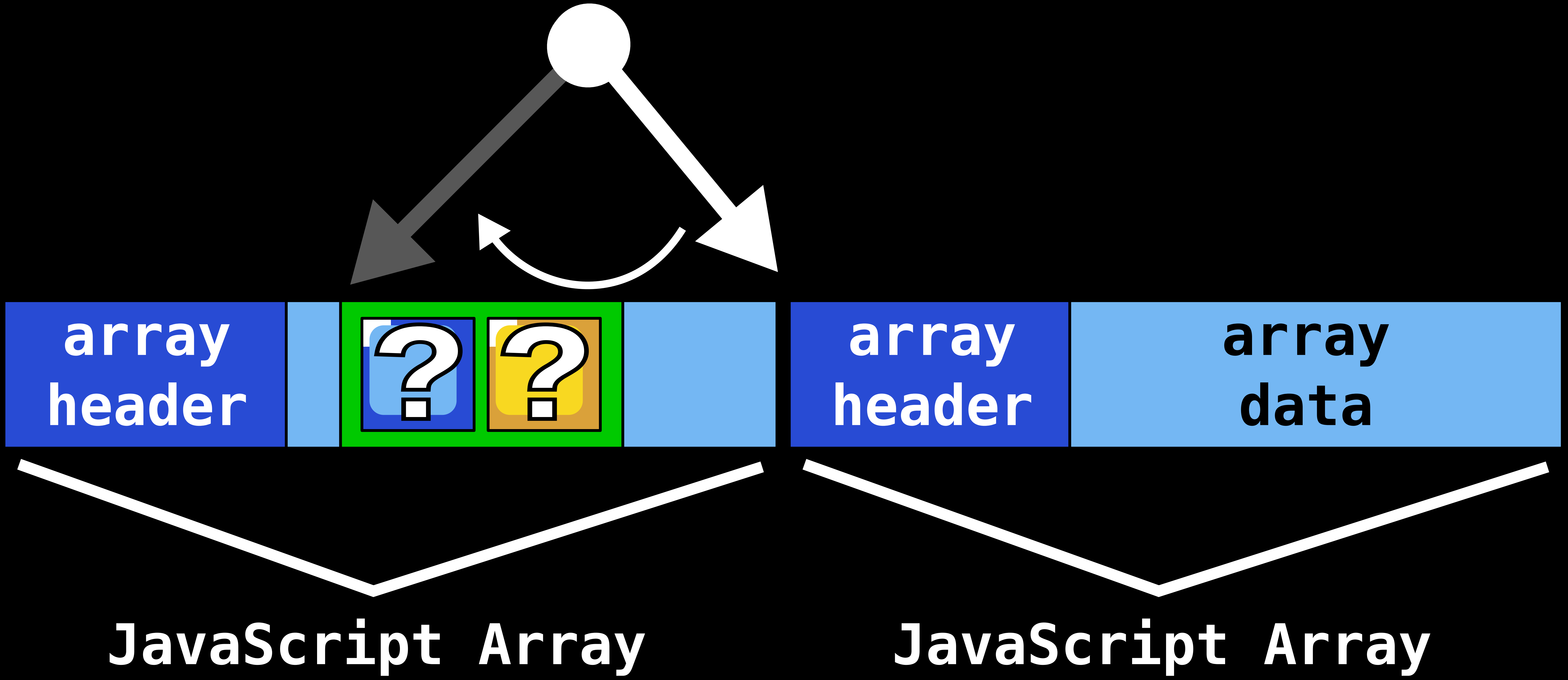
Pointer pivotting



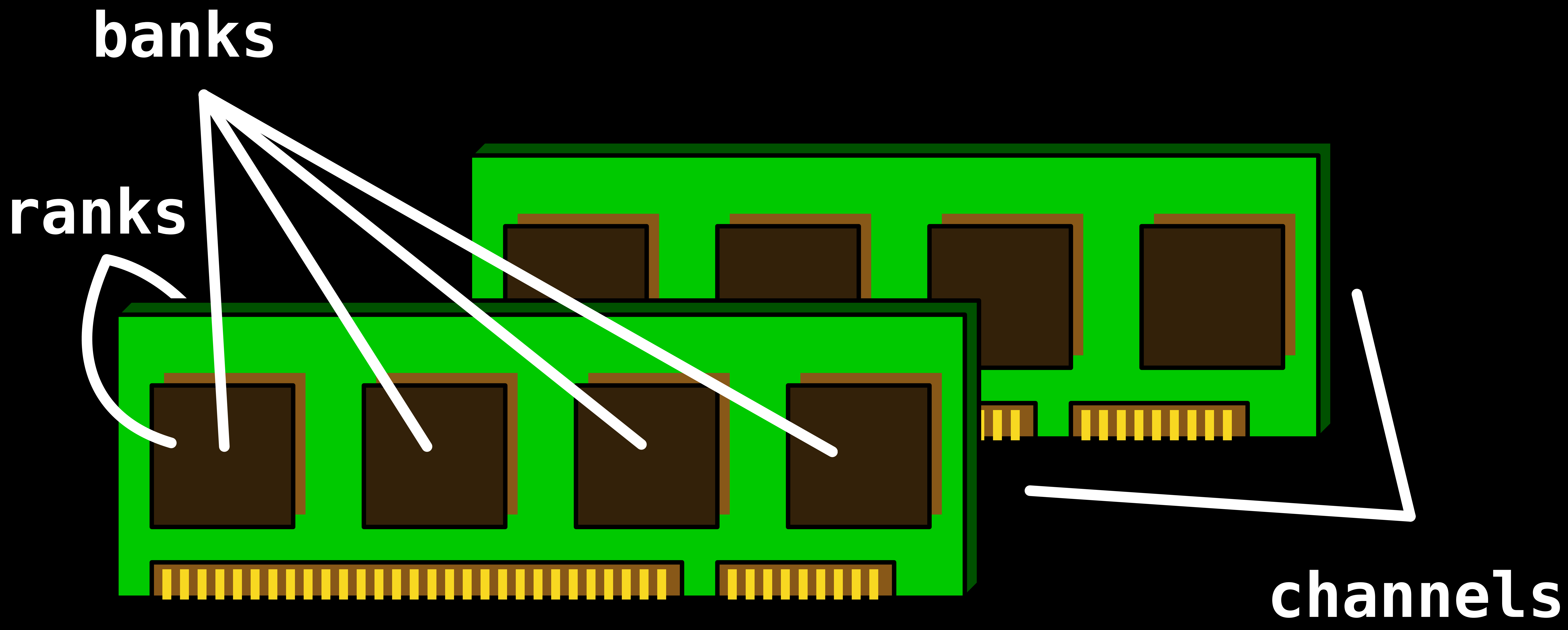
Pointer pivotting



Pointer pivotting

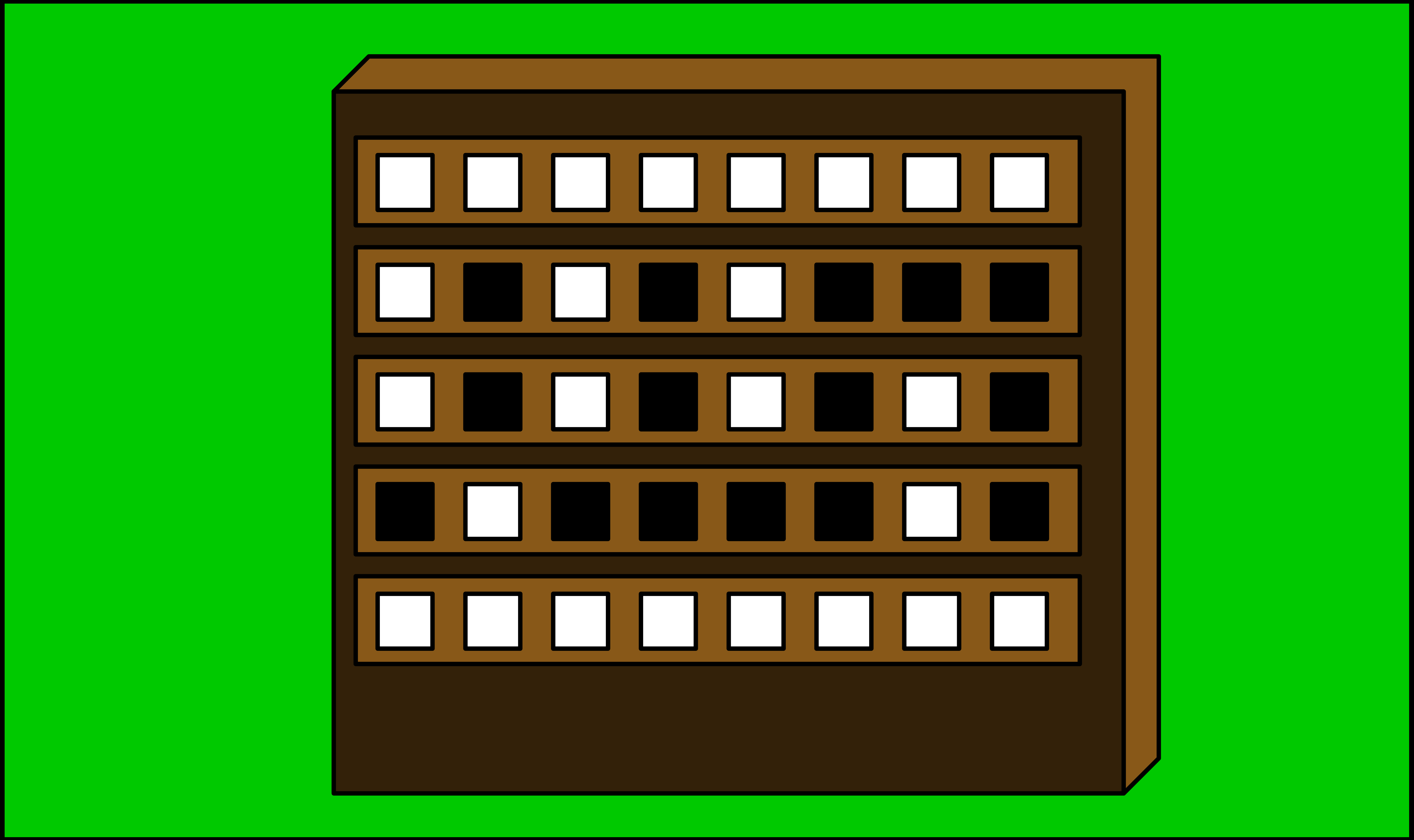


Rowhammer attack

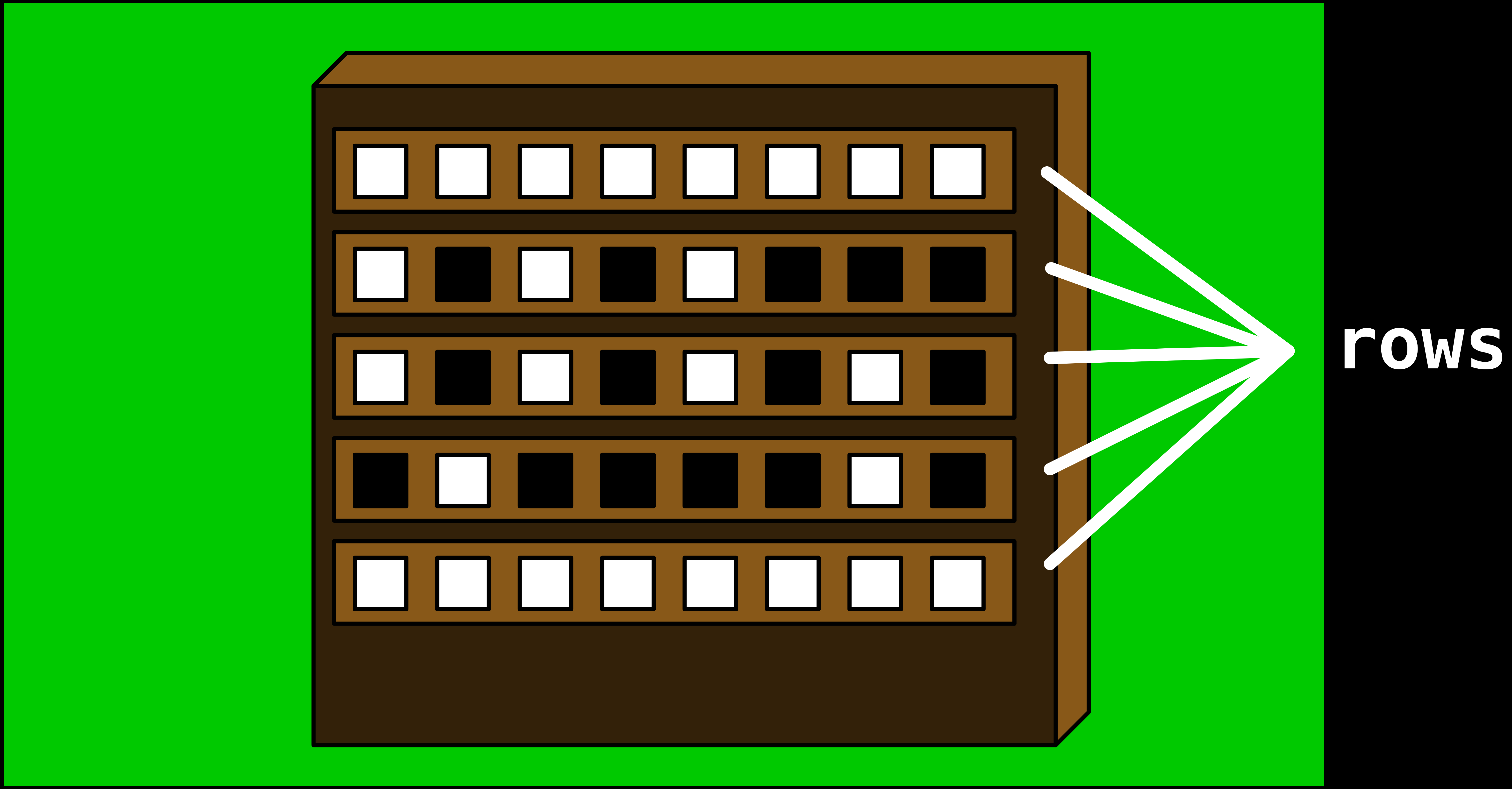


DDR memory

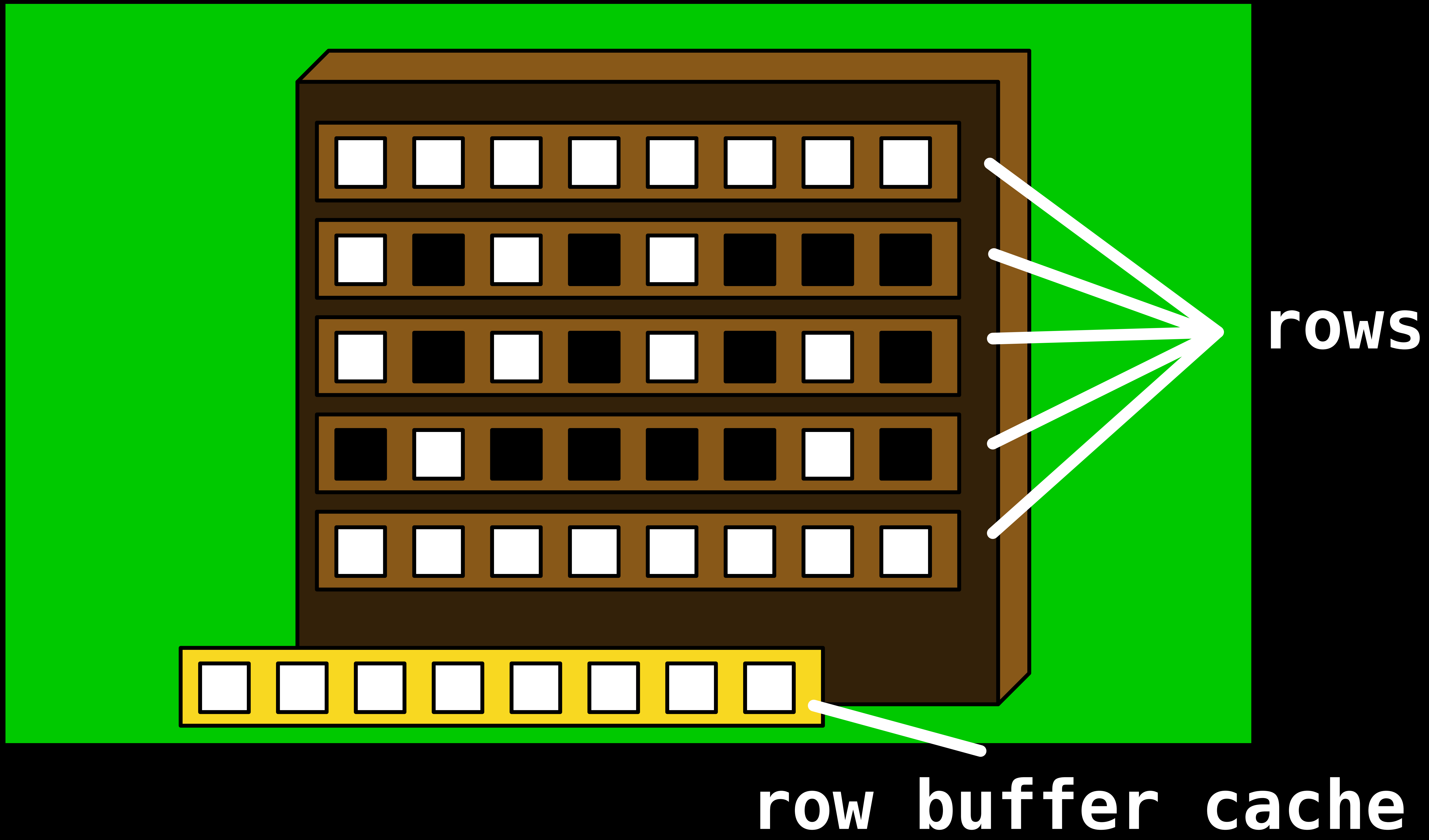
Rowhammer attack



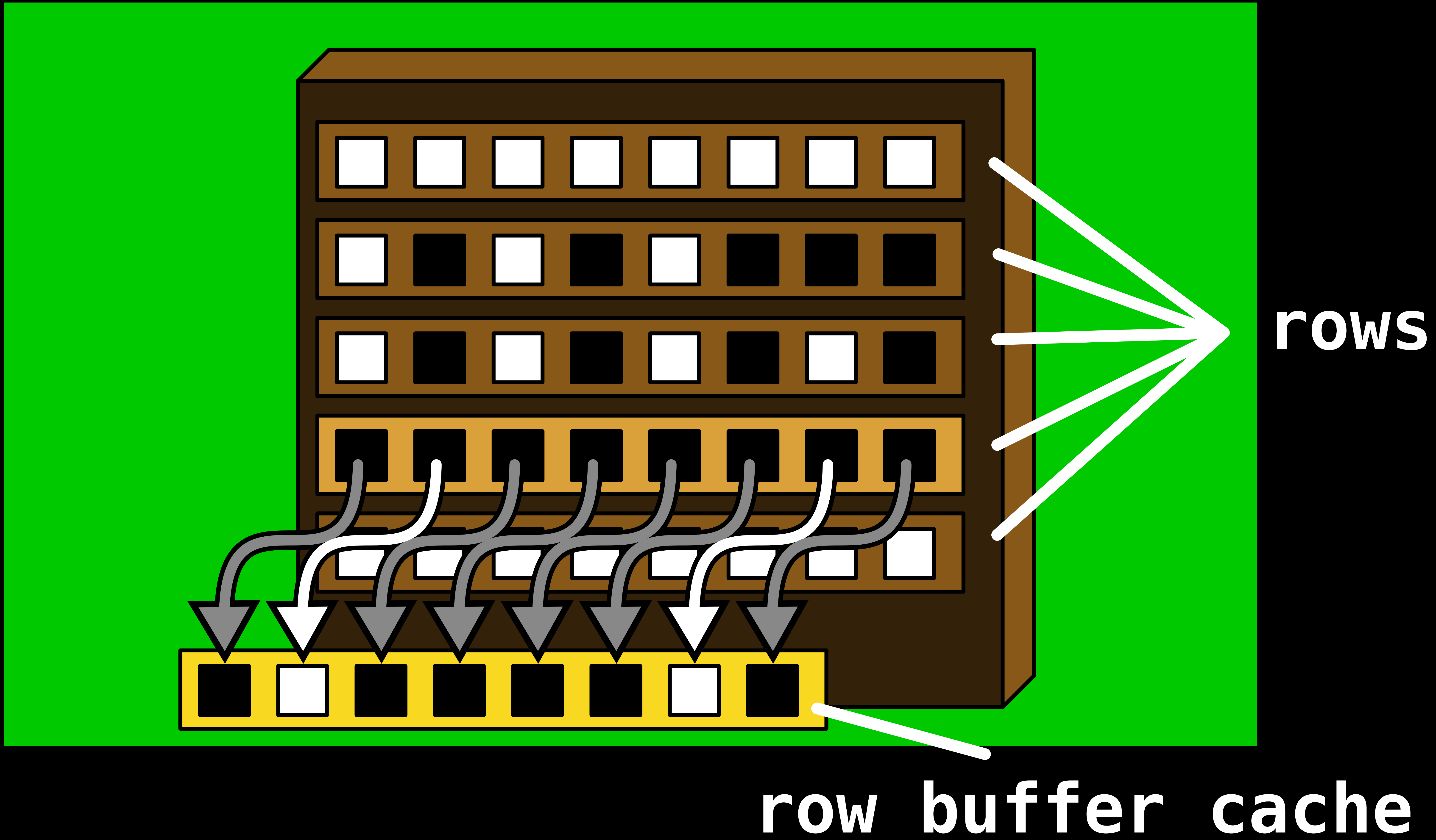
Rowhammer attack



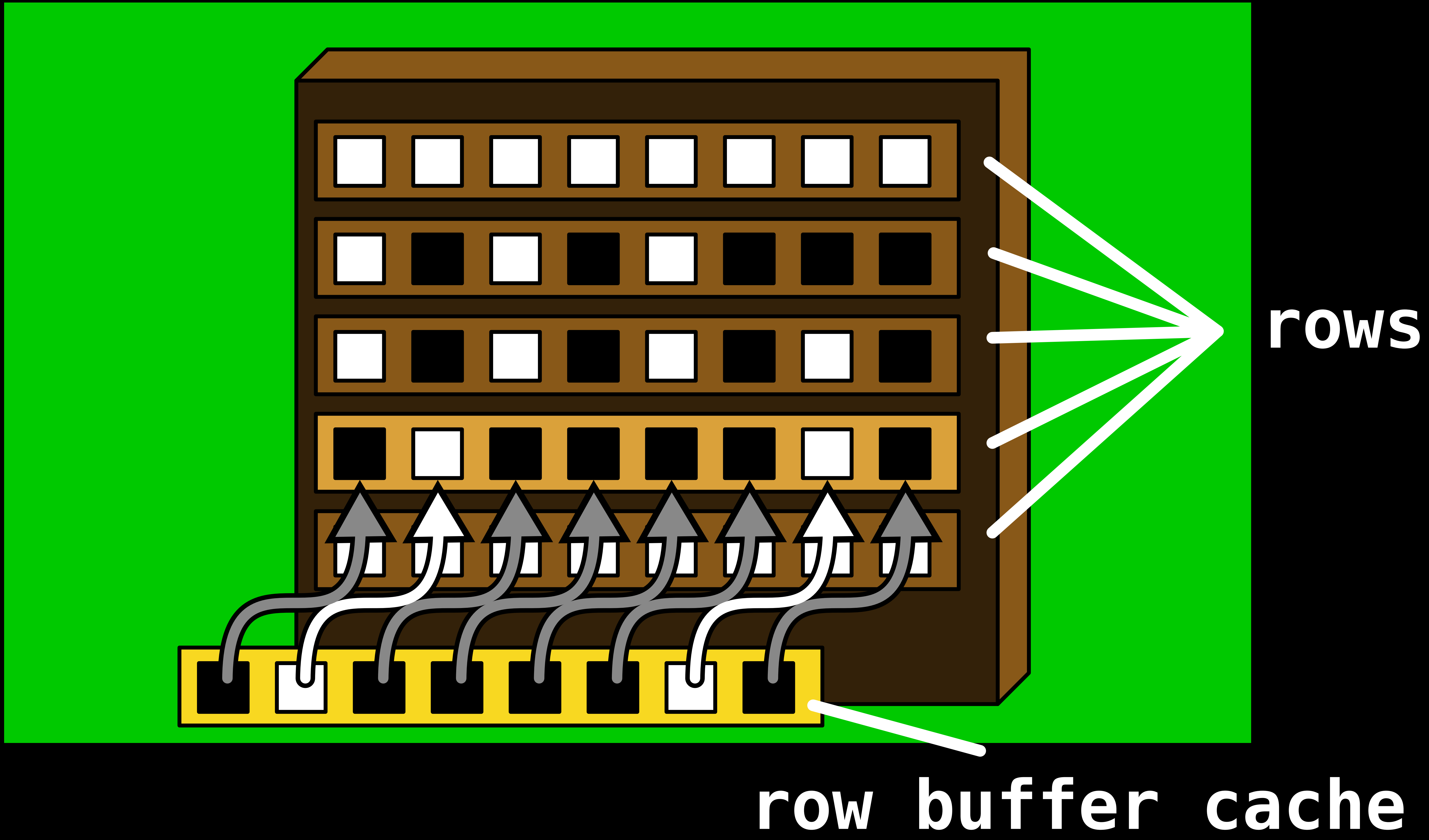
Rowhammer attack



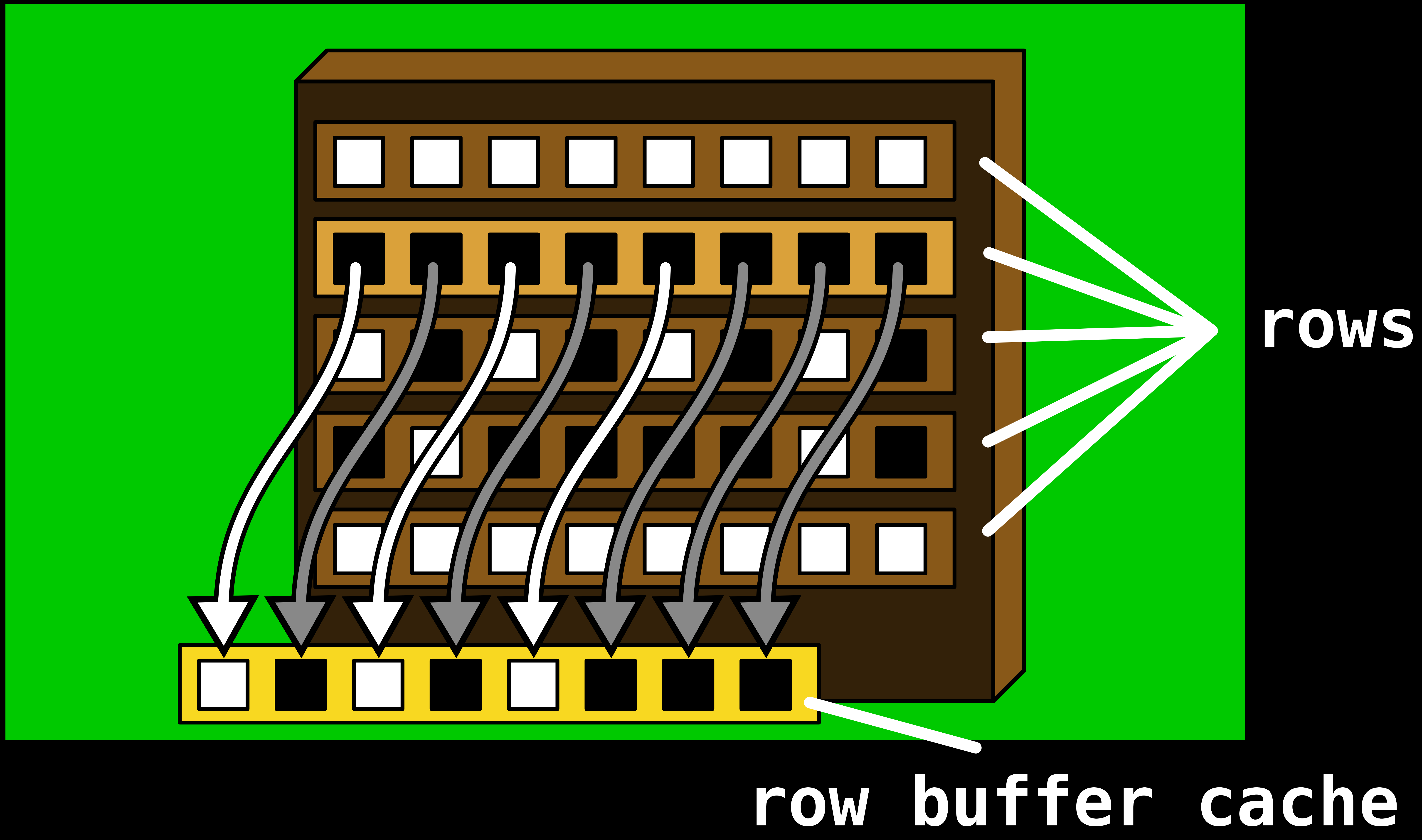
Rowhammer attack



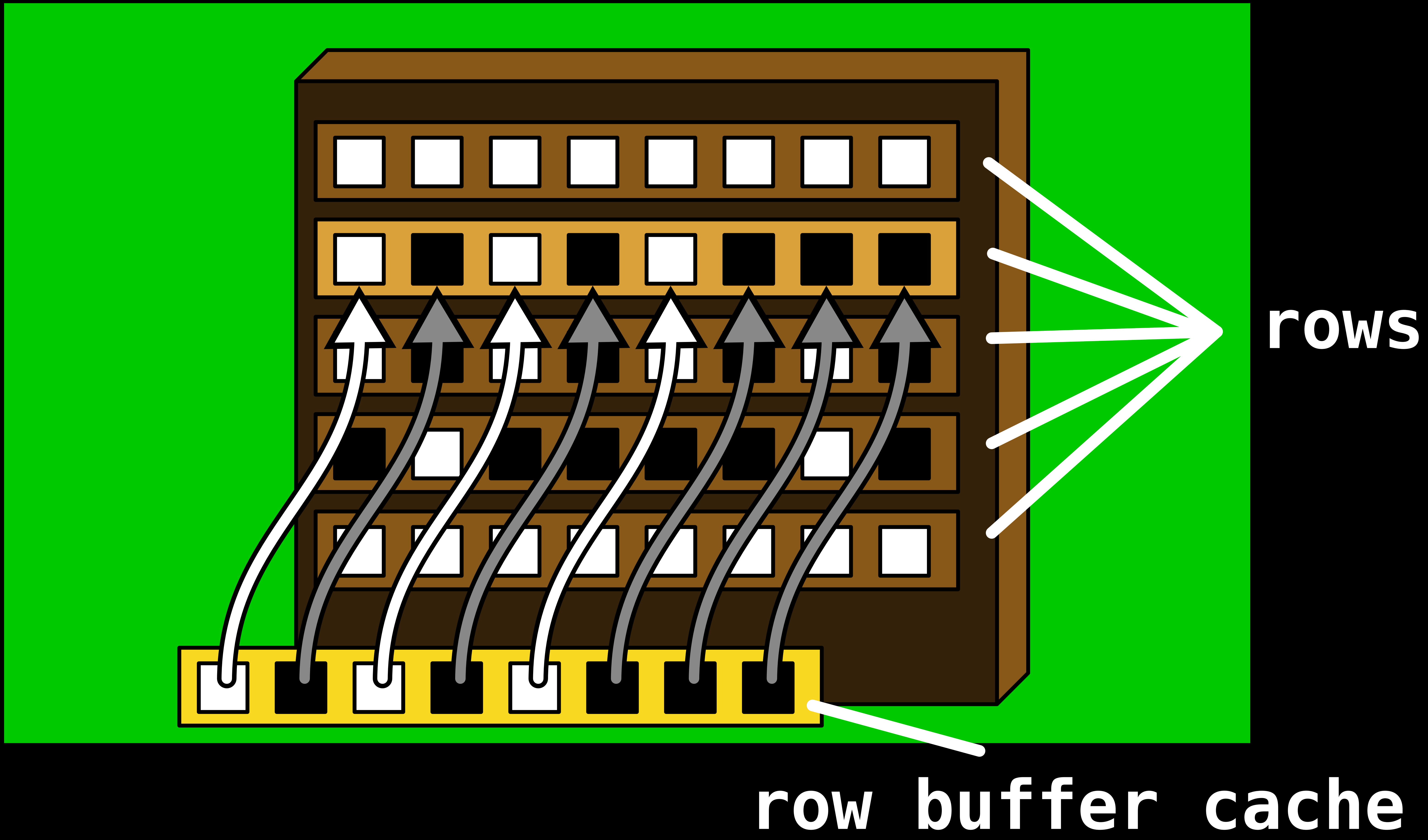
Rowhammer attack



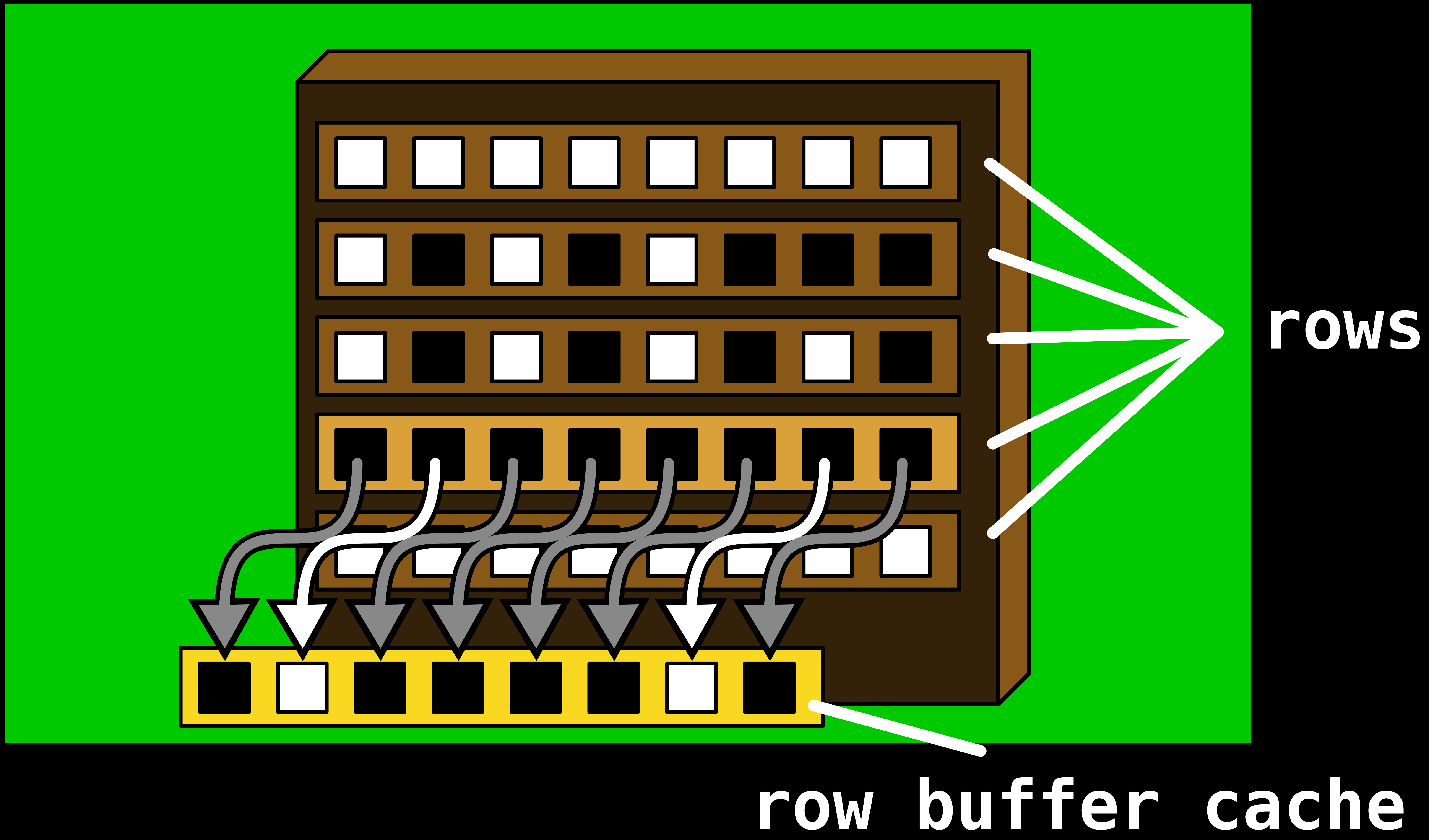
Rowhammer attack



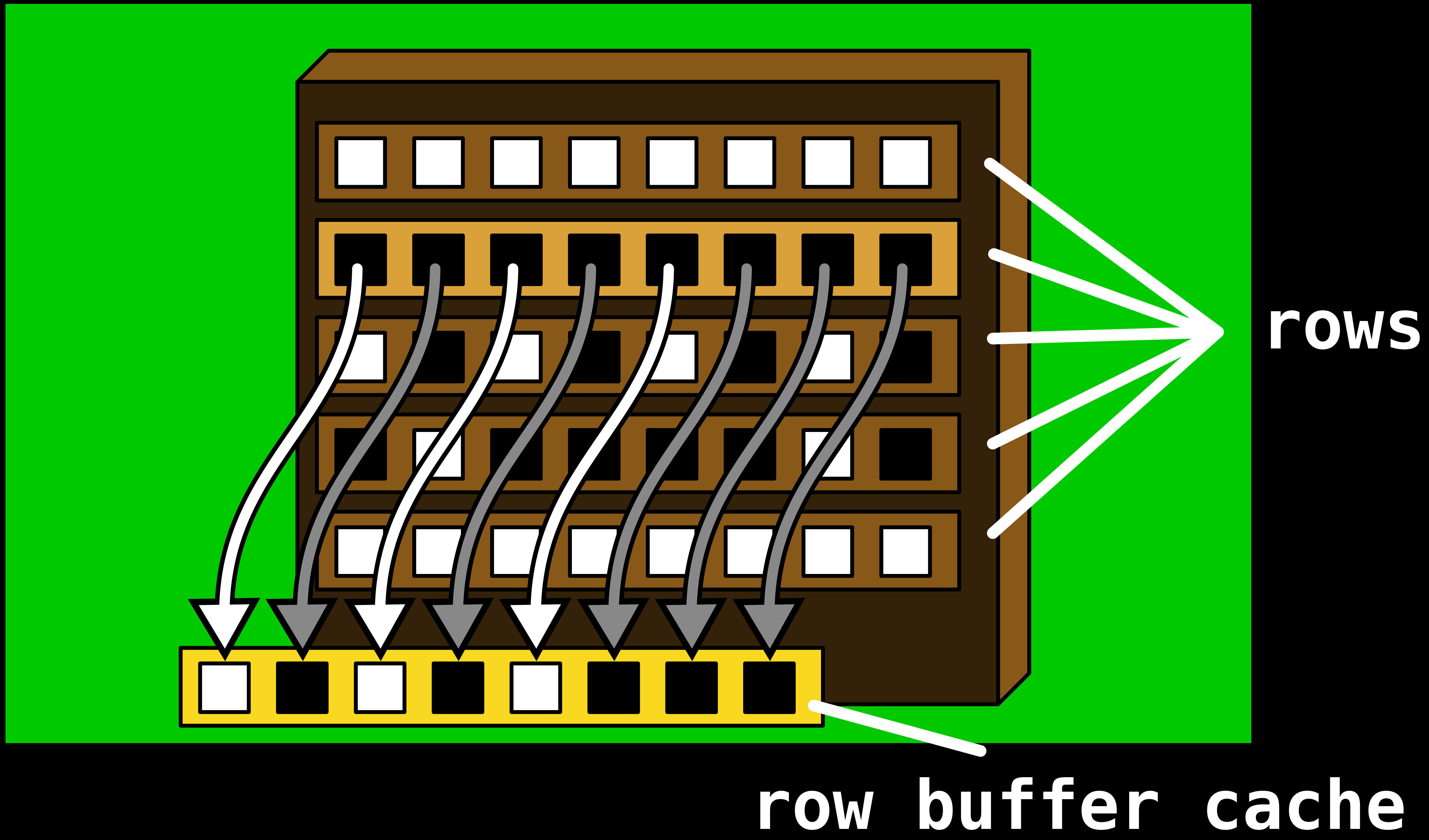
Rowhammer attack



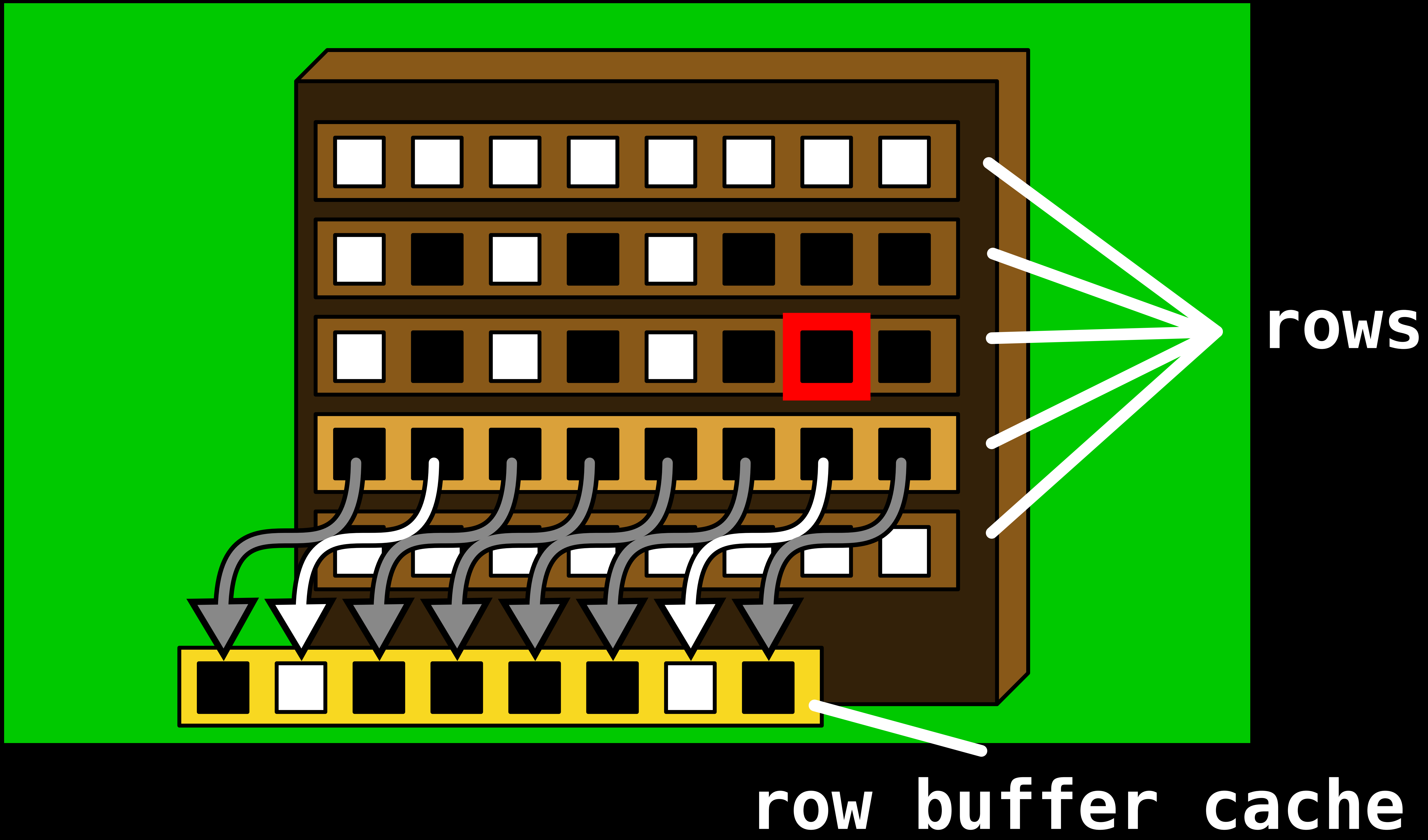
Rowhammer attack



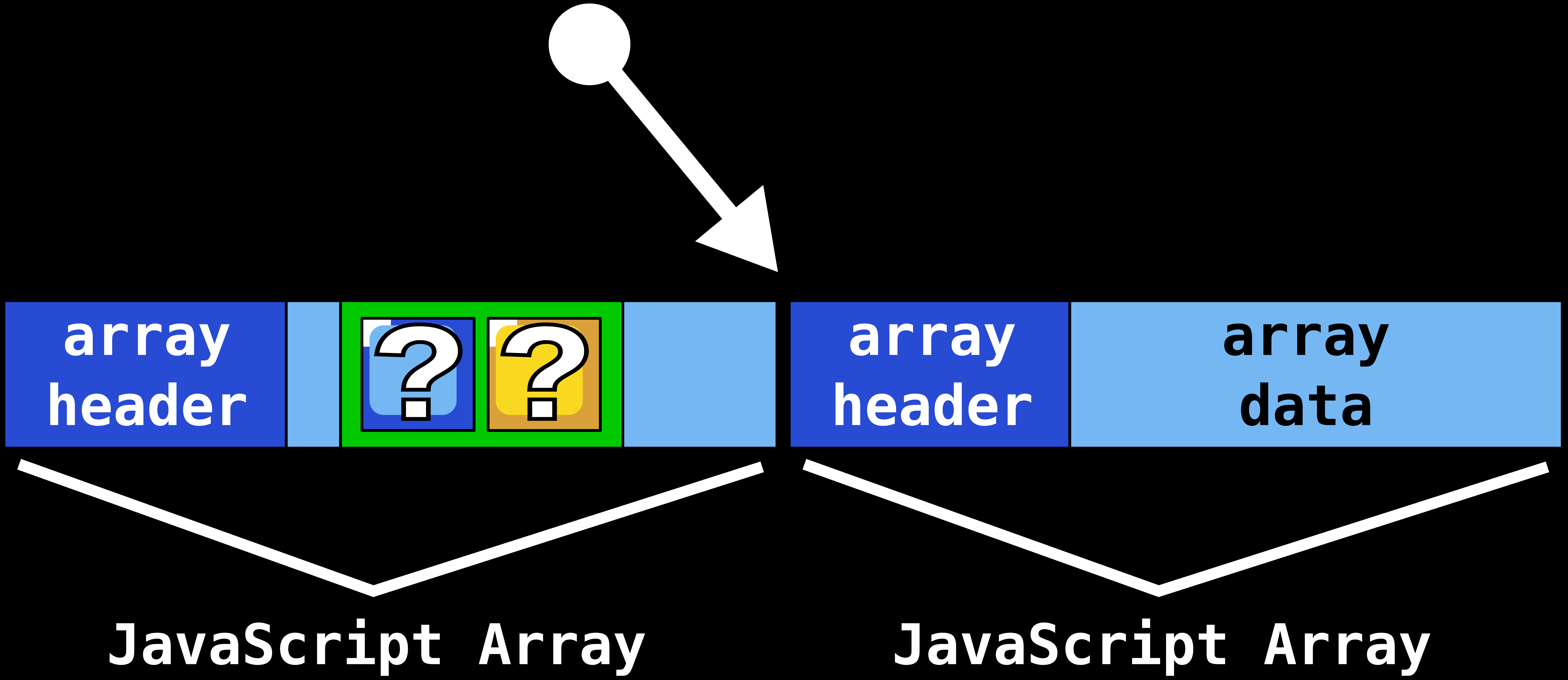
Rowhammer attack



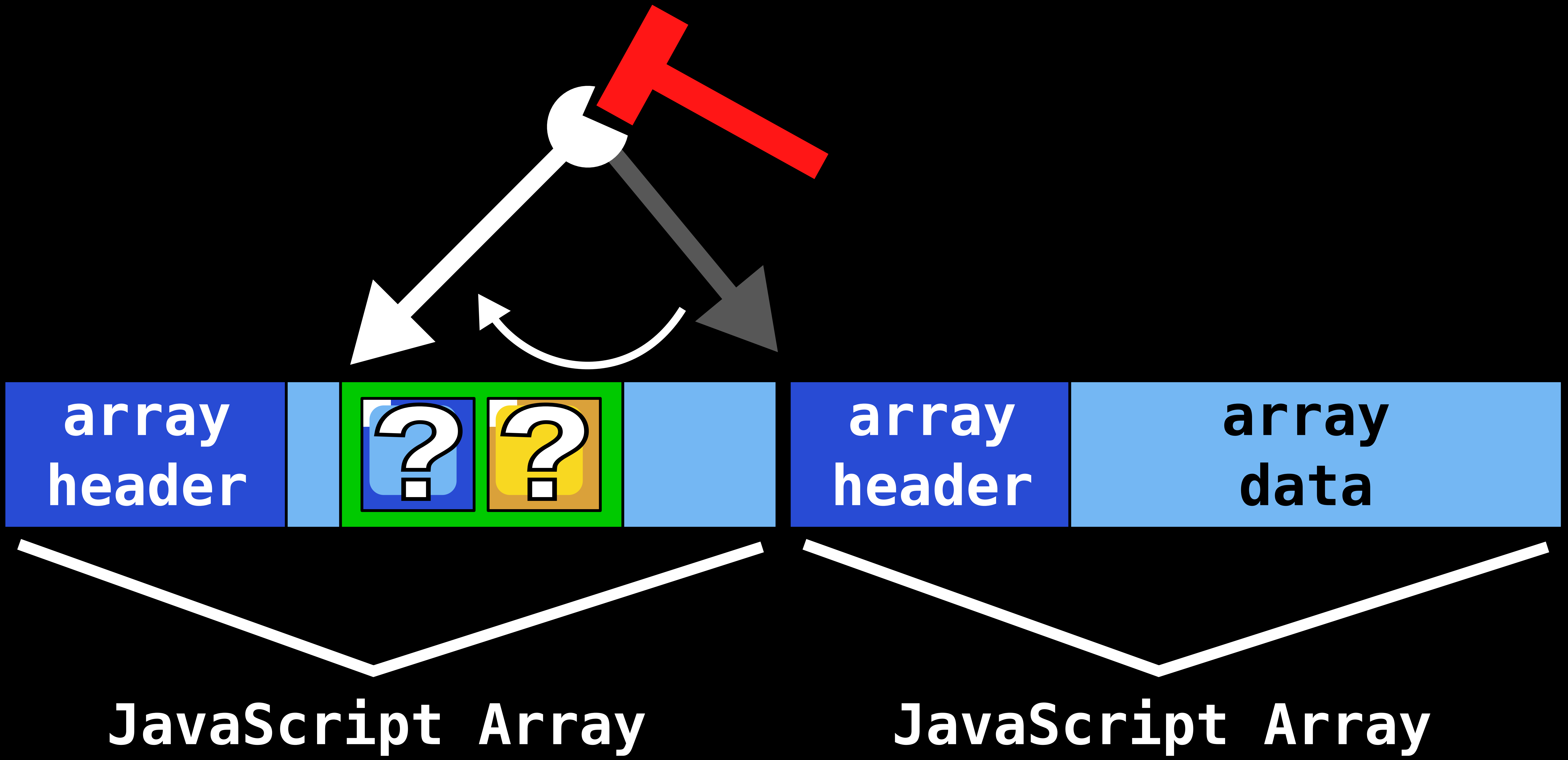
Rowhammer attack

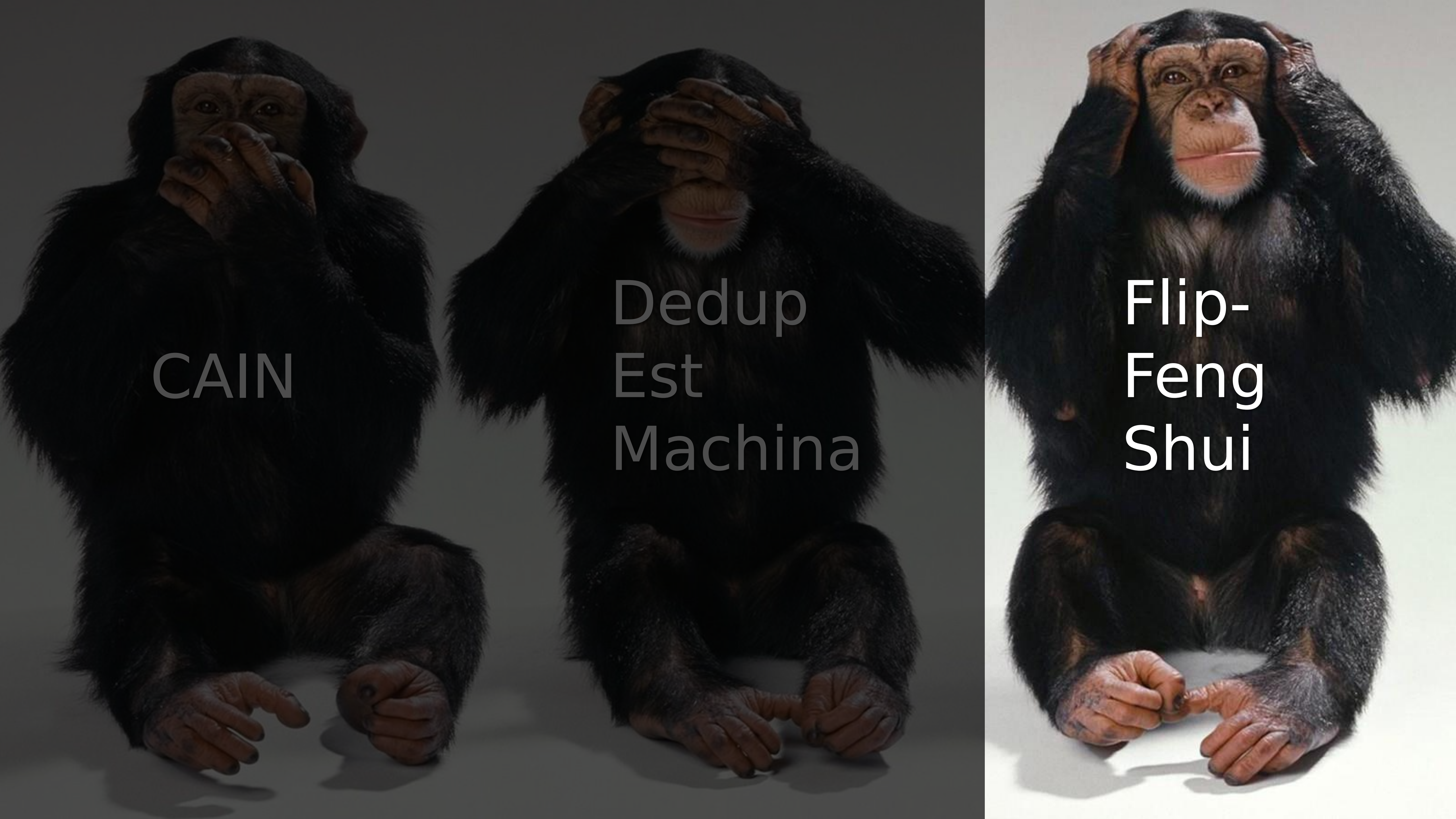


Pointer pivotting



Pointer pivotting




CAIN
Dedup
Est
Machina
Flip-
Feng
Shui

Flip Feng Shui

Rowhammer
(hardware bug)

Flip Feng Shui

Rowhammer
(hardware bug)
+
Deduplication
(more than a software side-channel)

Flip Feng Shui

Rowhammer
(hardware bug)
+
Deduplication
(more than a software side-channel)

↓

Cross-VM compromise

Rowhammer bit flips:

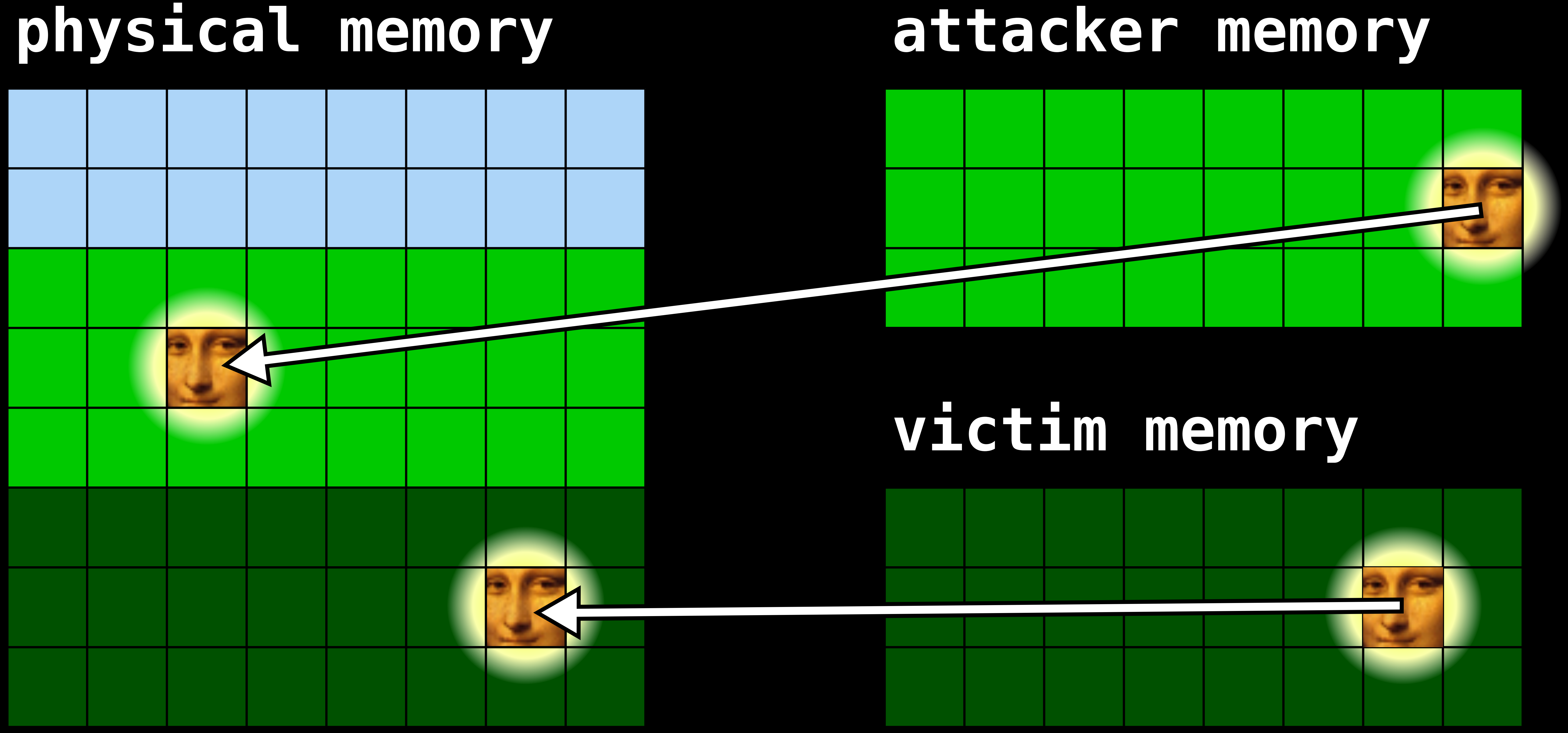
1) Unpredictable on which (virtual) page
2) Unpredictable where in the page
3) Repeatable once you've found a flip

Flip Feng Shui goal:

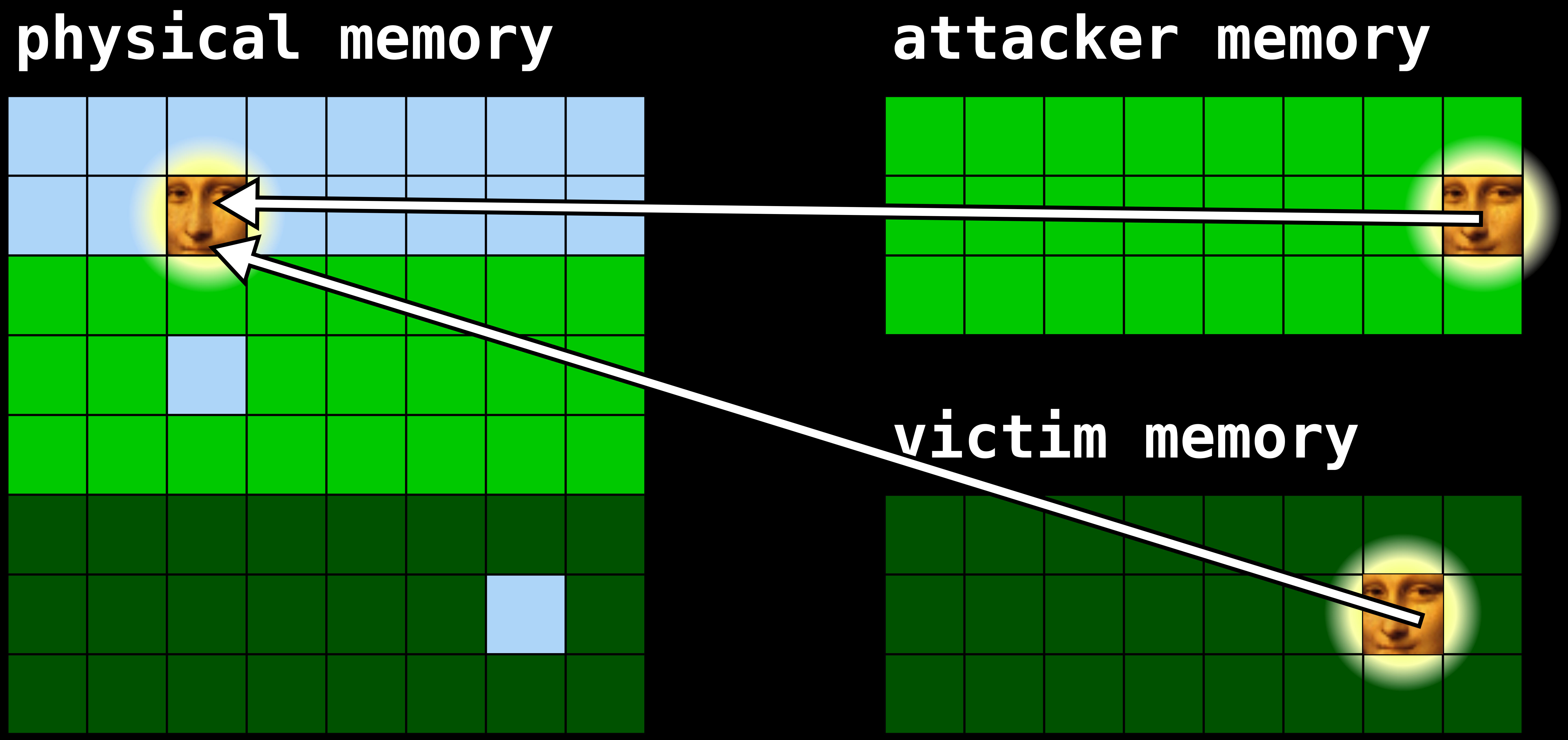
> Find victim pages with known content which allow for exploitation when certain bits are flipped

> Land this victim page in a physical memory location where this bit is flippable

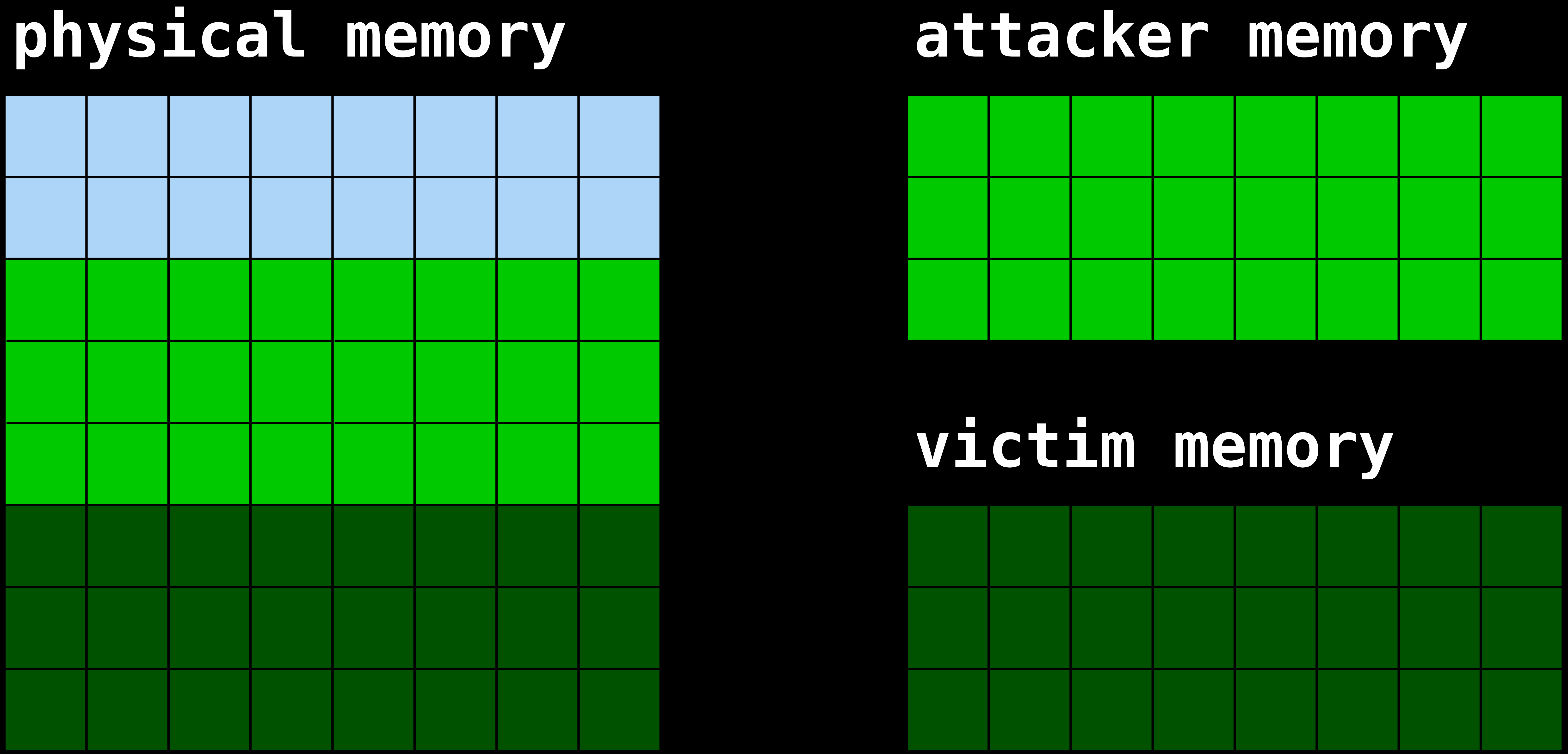
Deduplication implementation: Windows 10



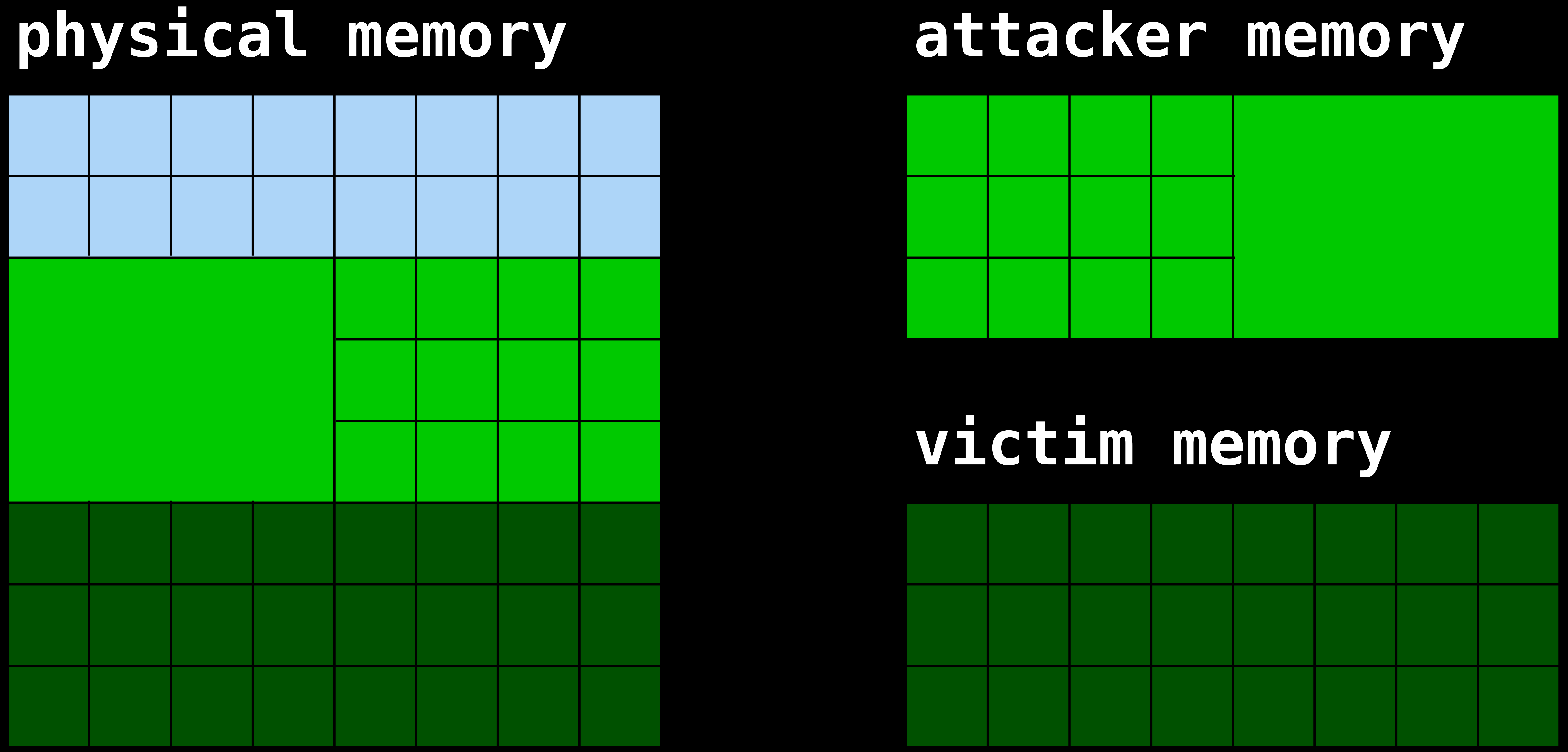
Deduplication implementation: Windows 10



Deduplication implementation: KVM on Linux (KSM)

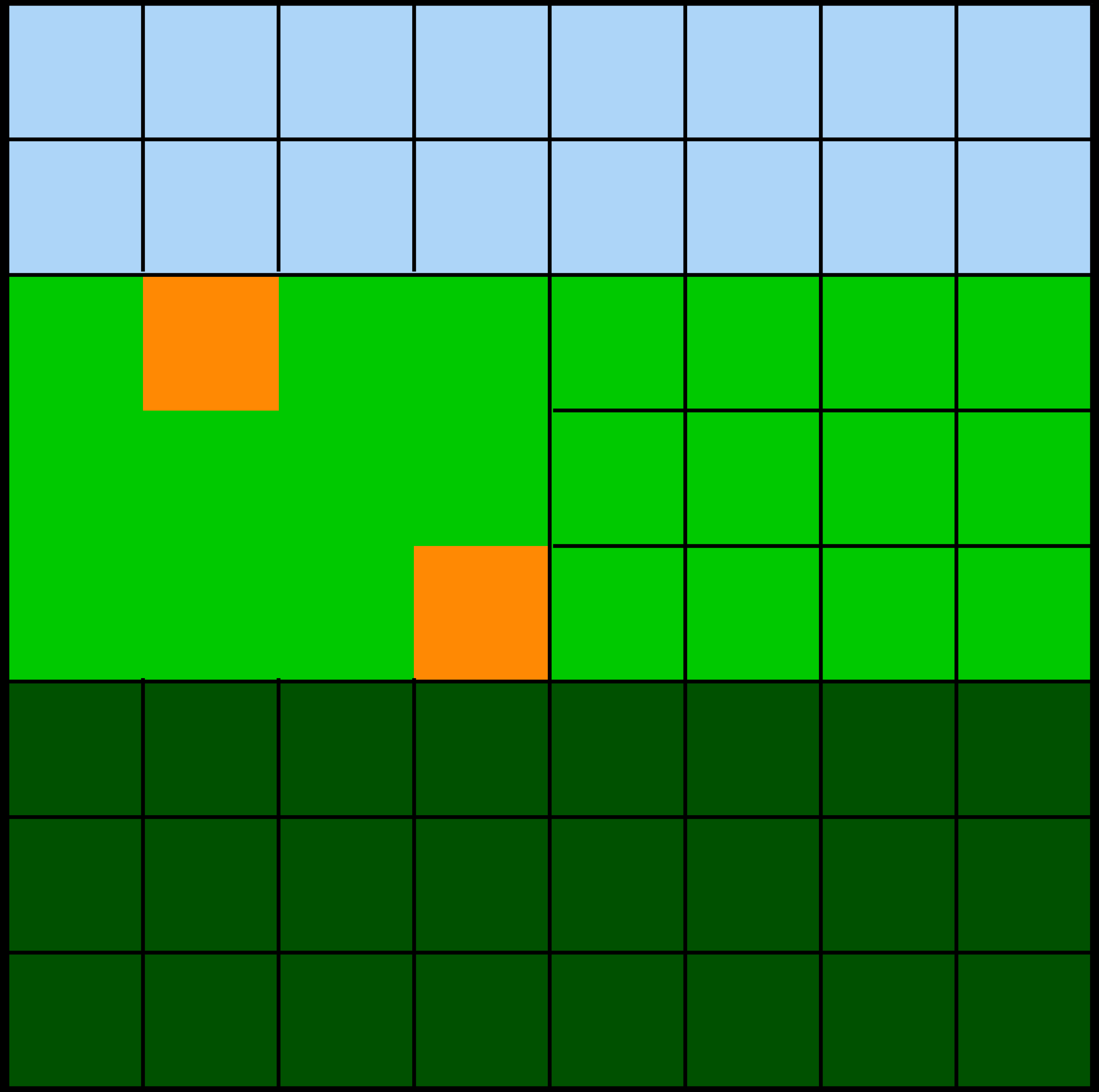


Deduplication implementation: KVM on Linux (KSM)

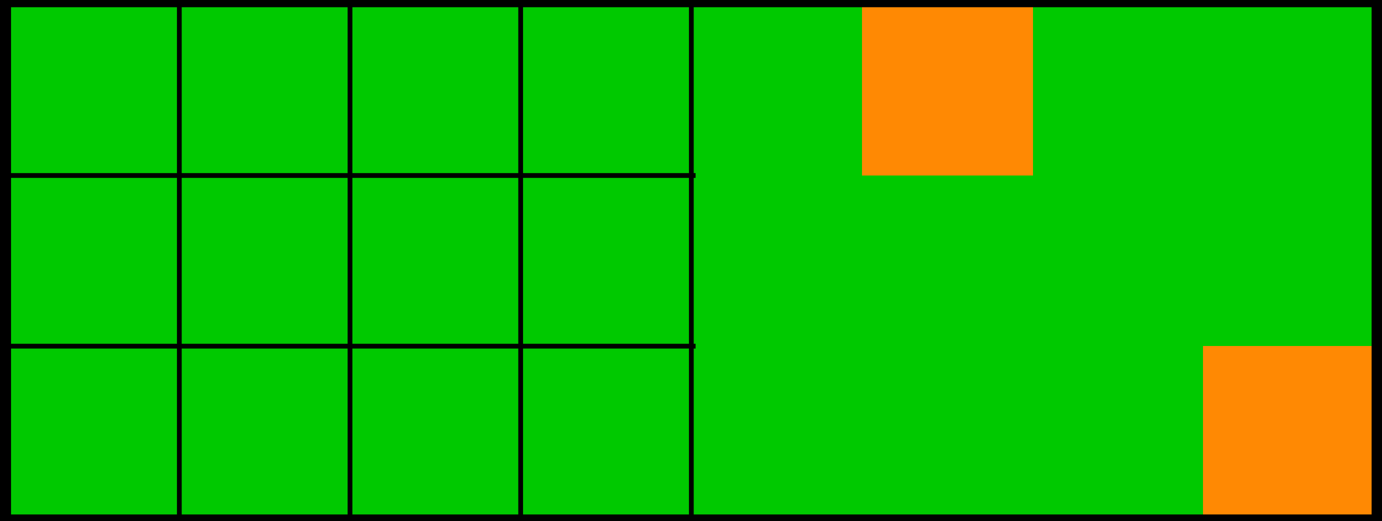


Deduplication implementation:
KVM on Linux (KSM)

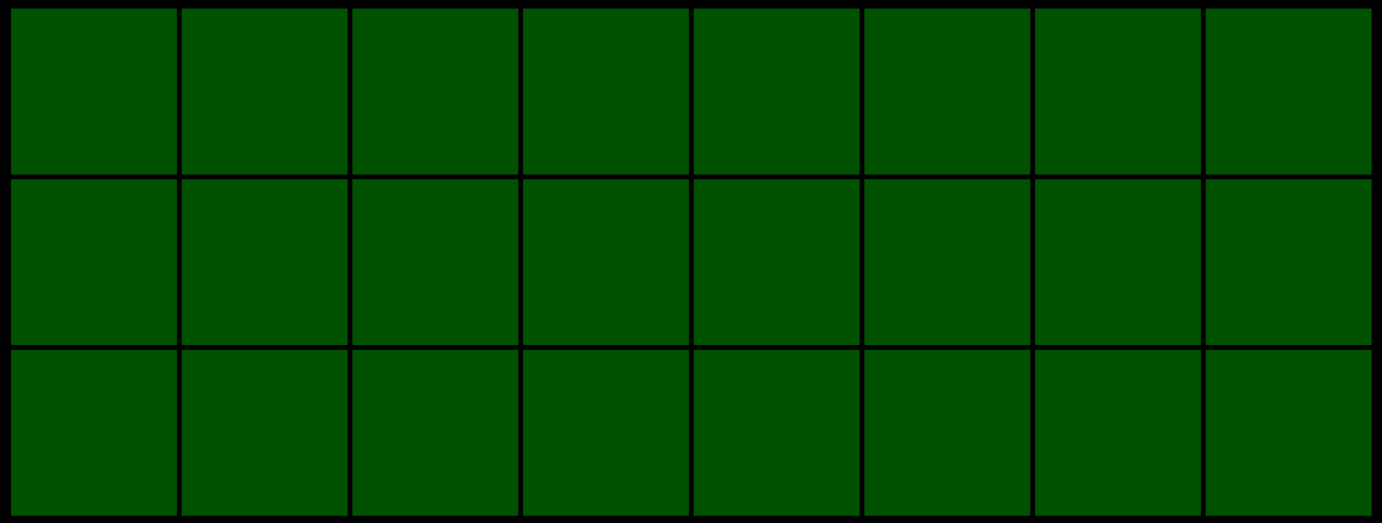
physical memory



attacker memory

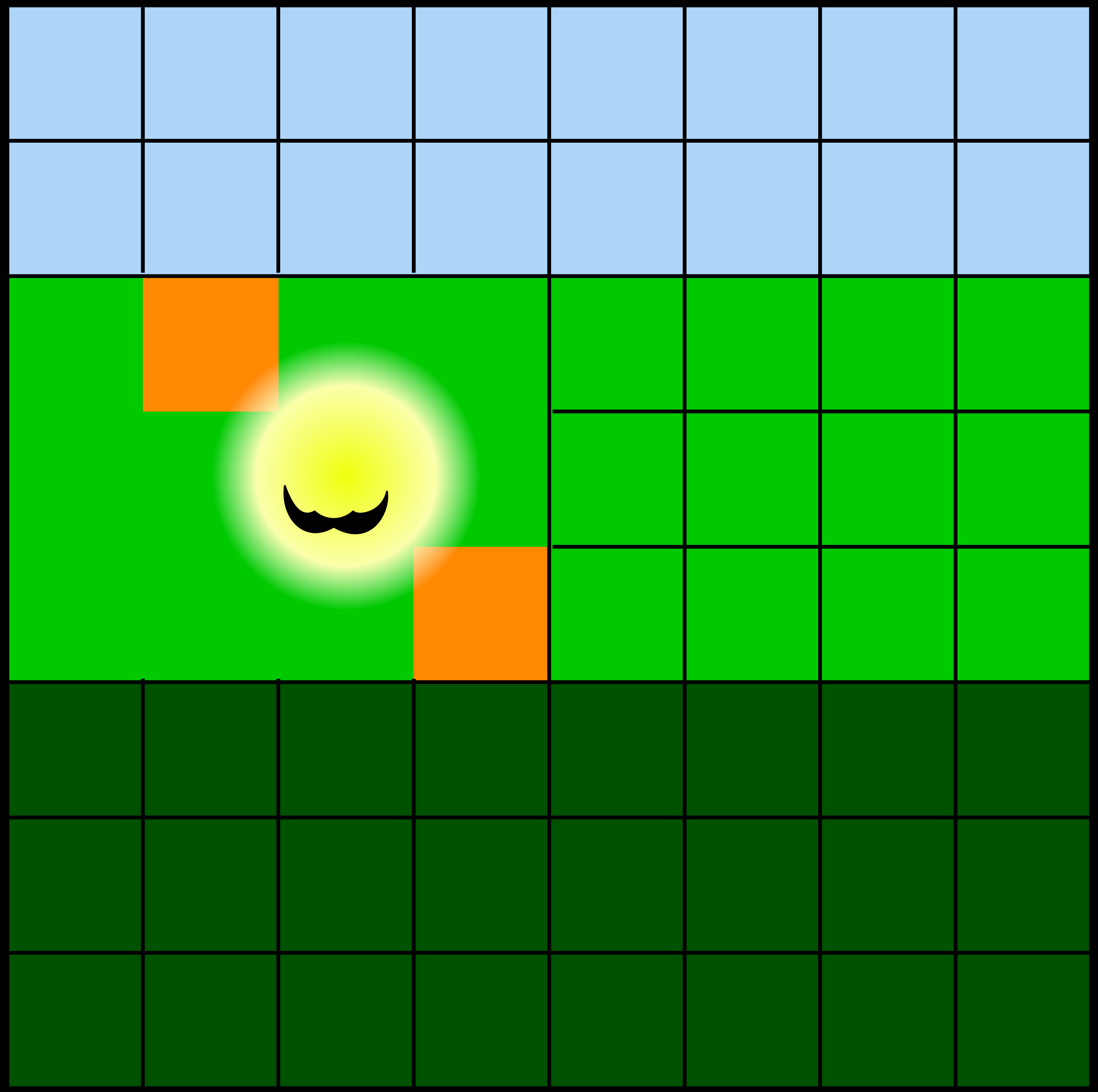


victim memory

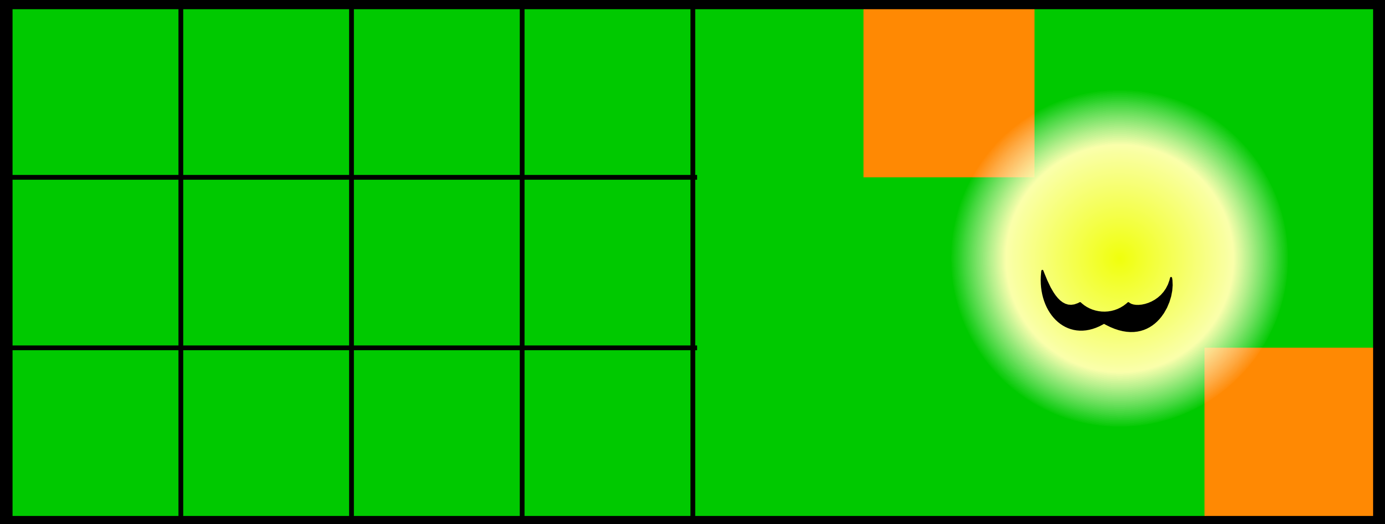


Deduplication implementation: KVM on Linux (KSM)

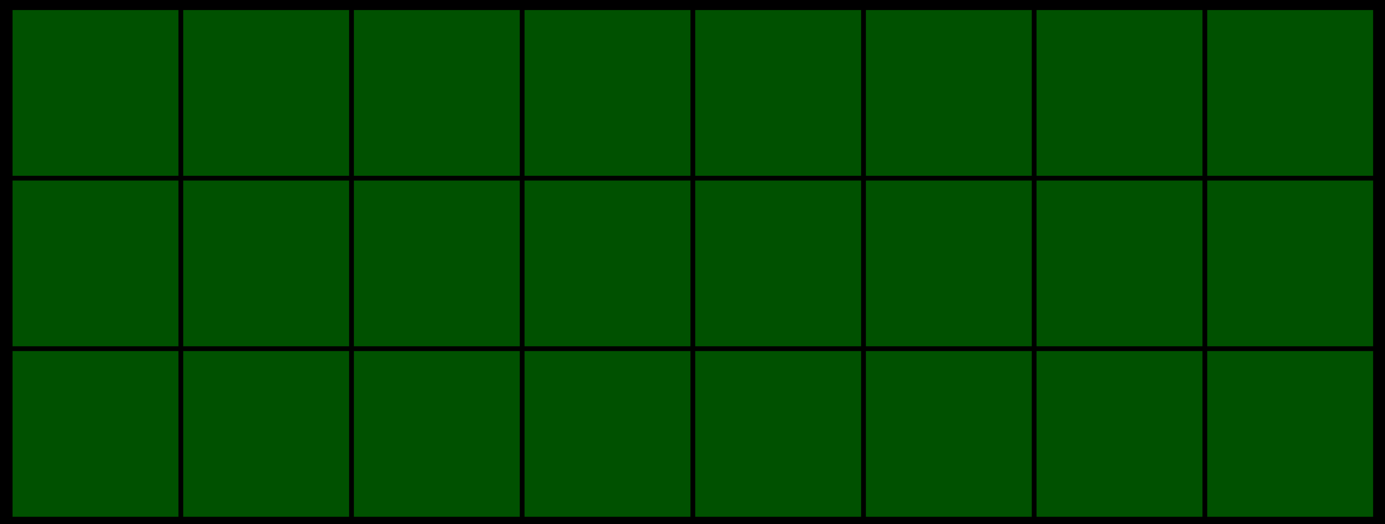
physical memory



attacker memory

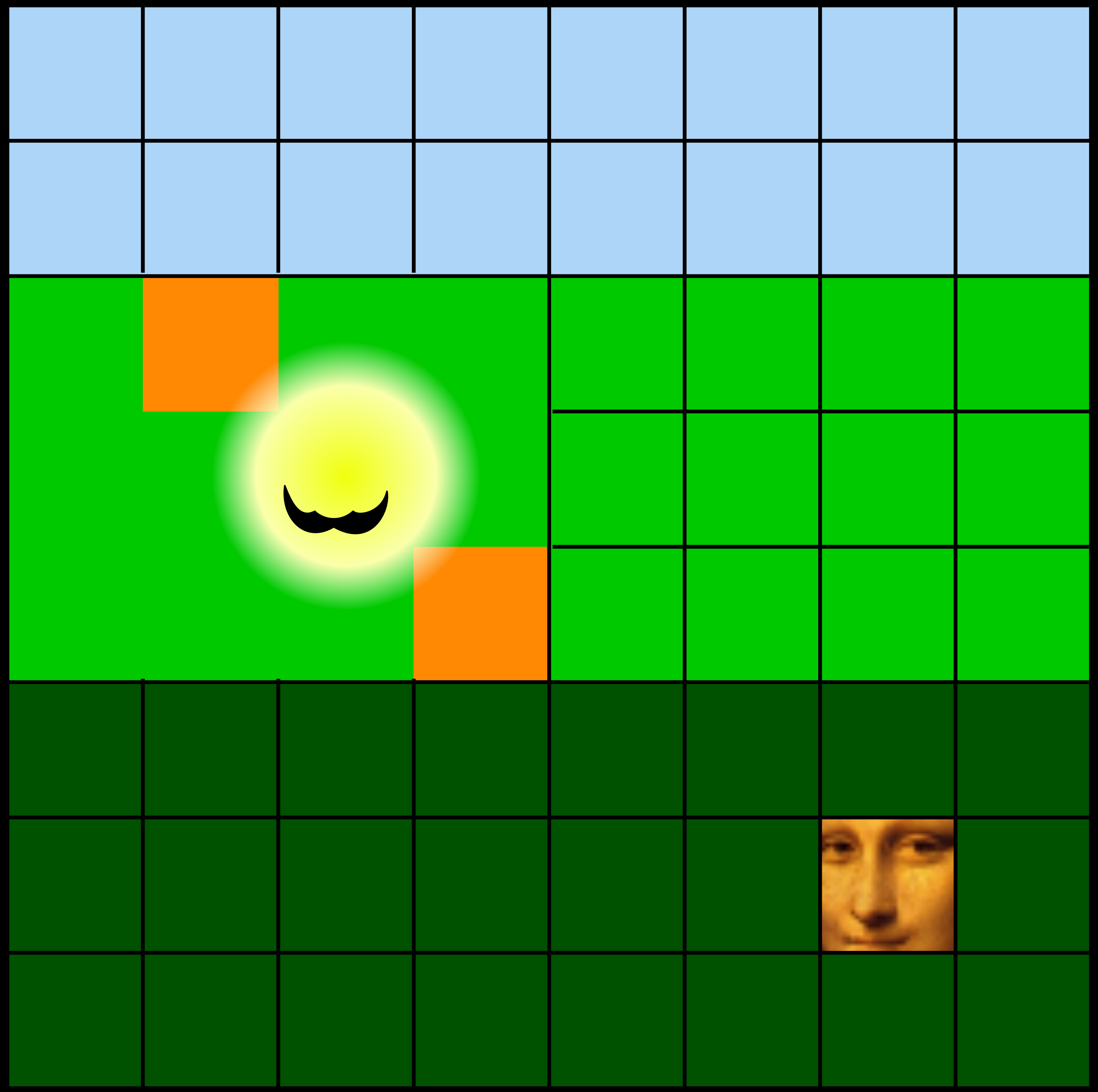


victim memory

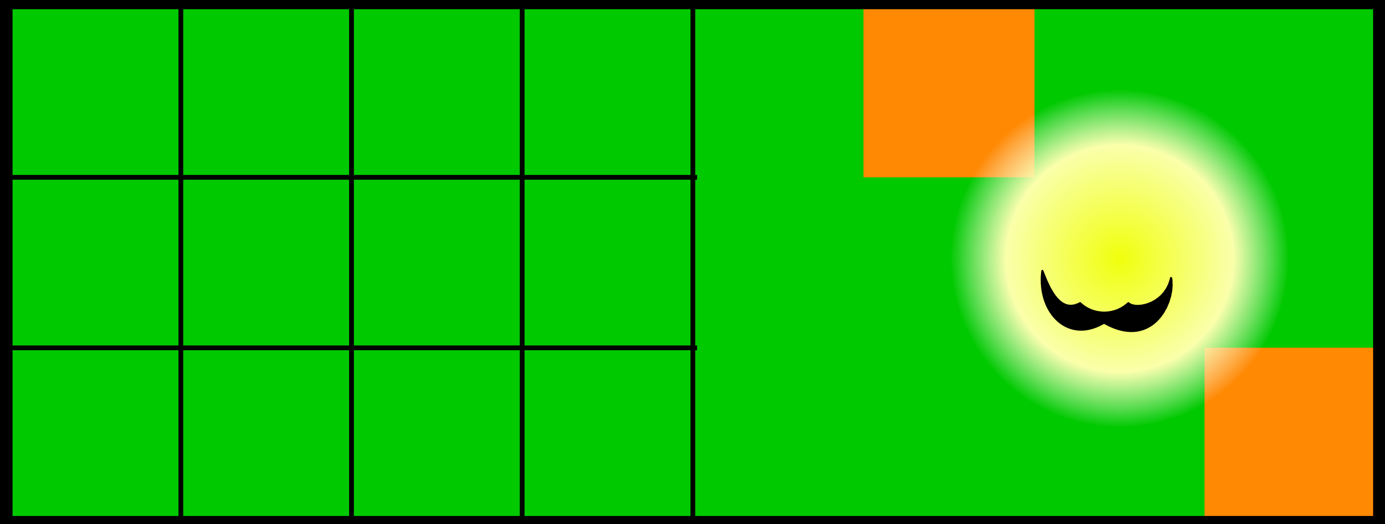


Deduplication implementation: KVM on Linux (KSM)

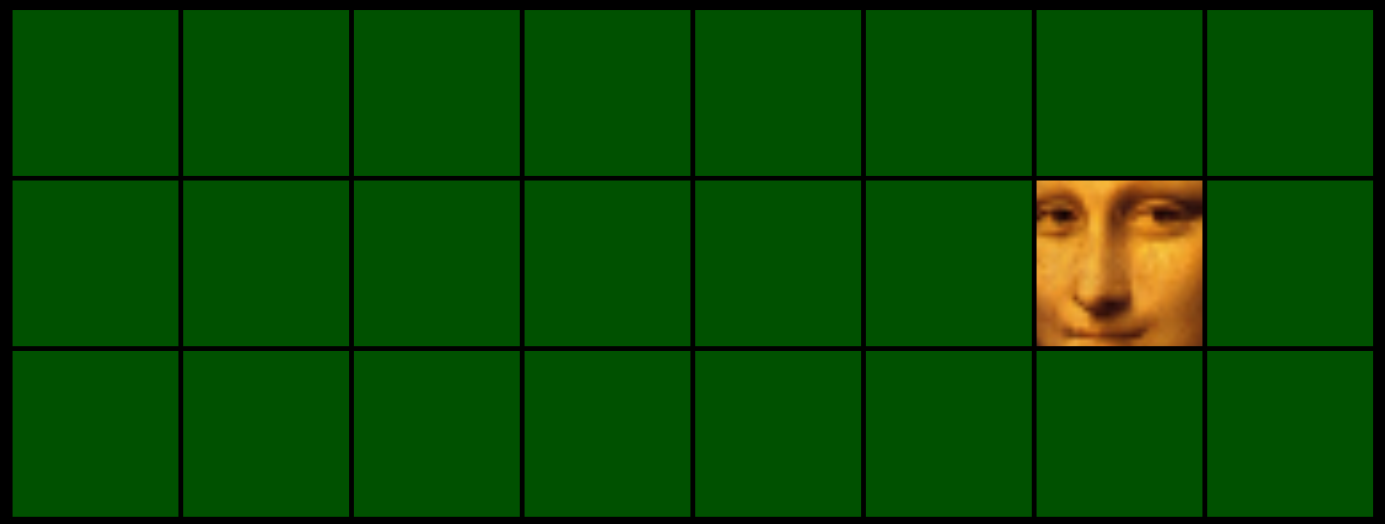
physical memory



attacker memory

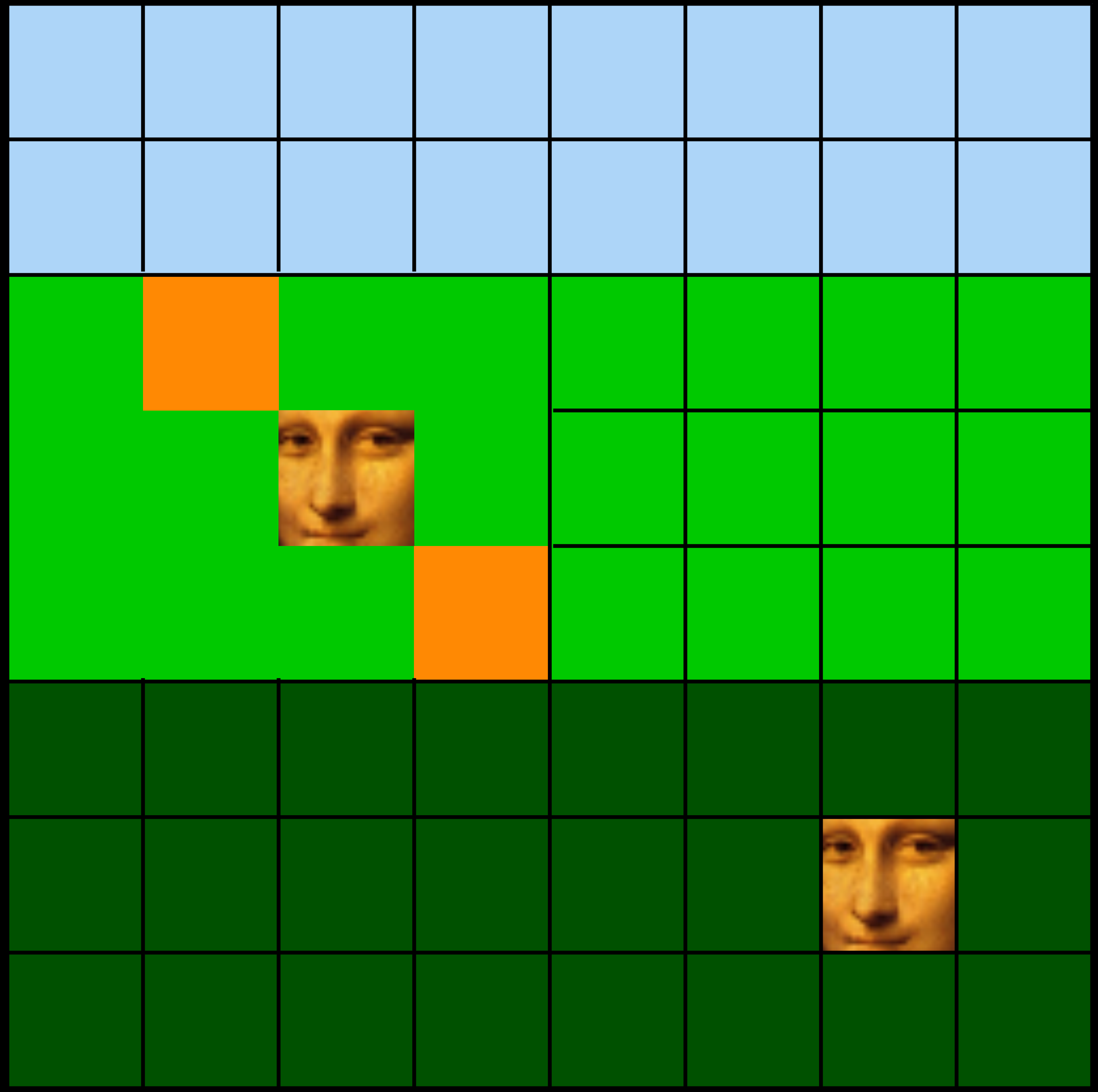


victim memory

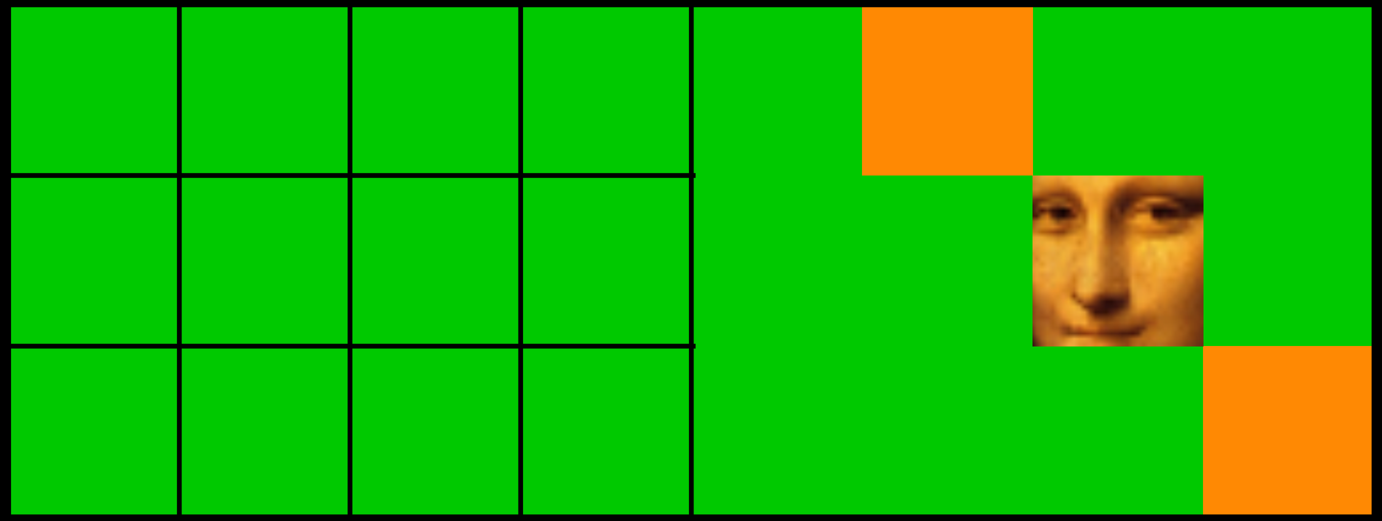


Deduplication implementation: KVM on Linux (KSM)

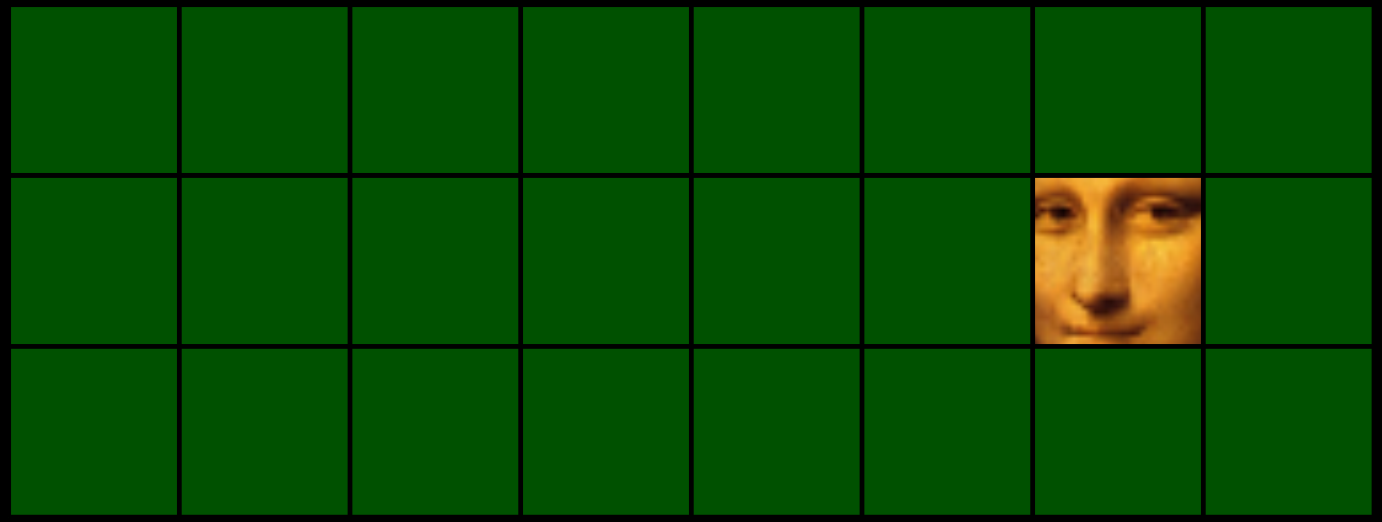
physical memory



attacker memory

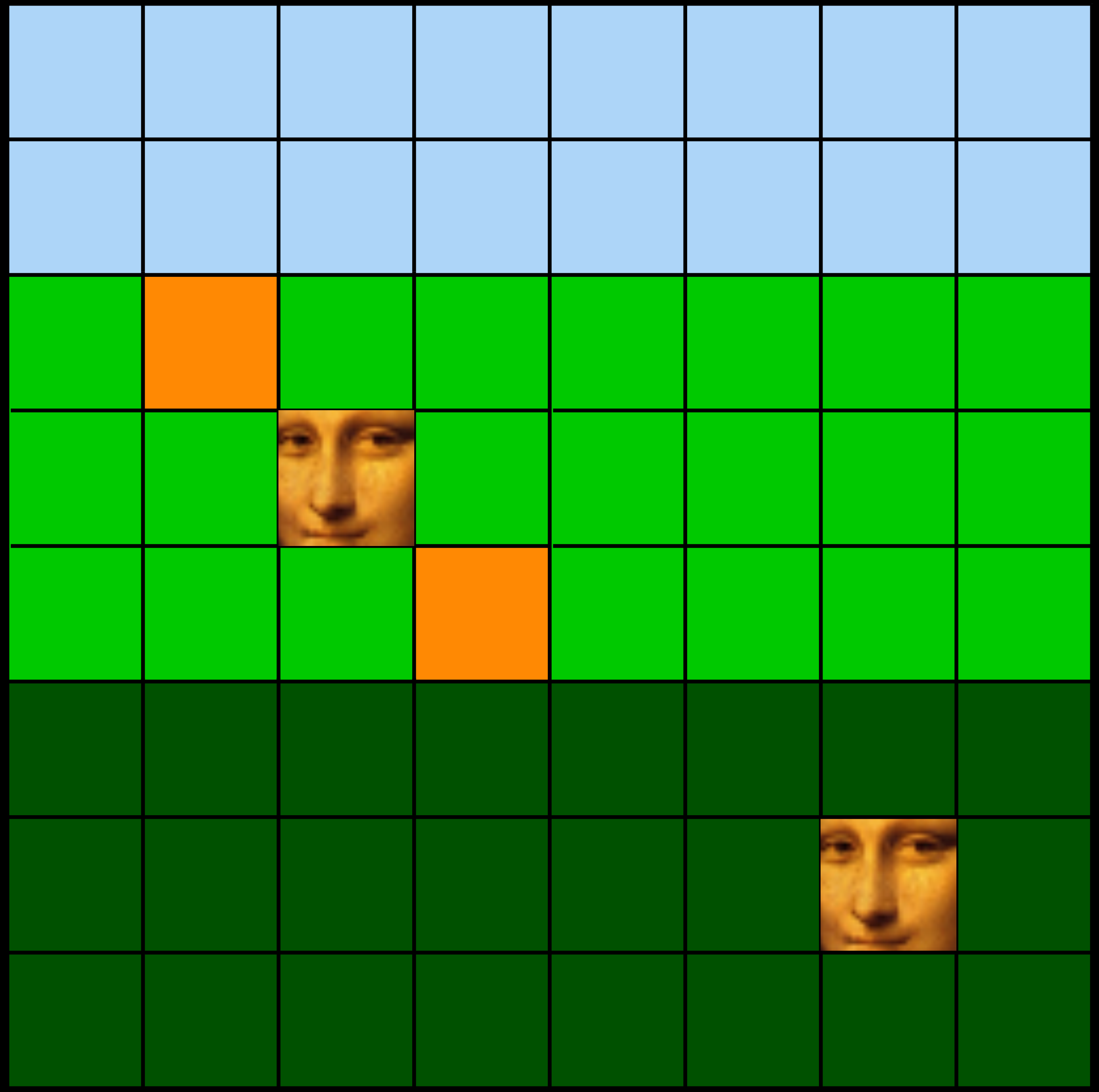


victim memory

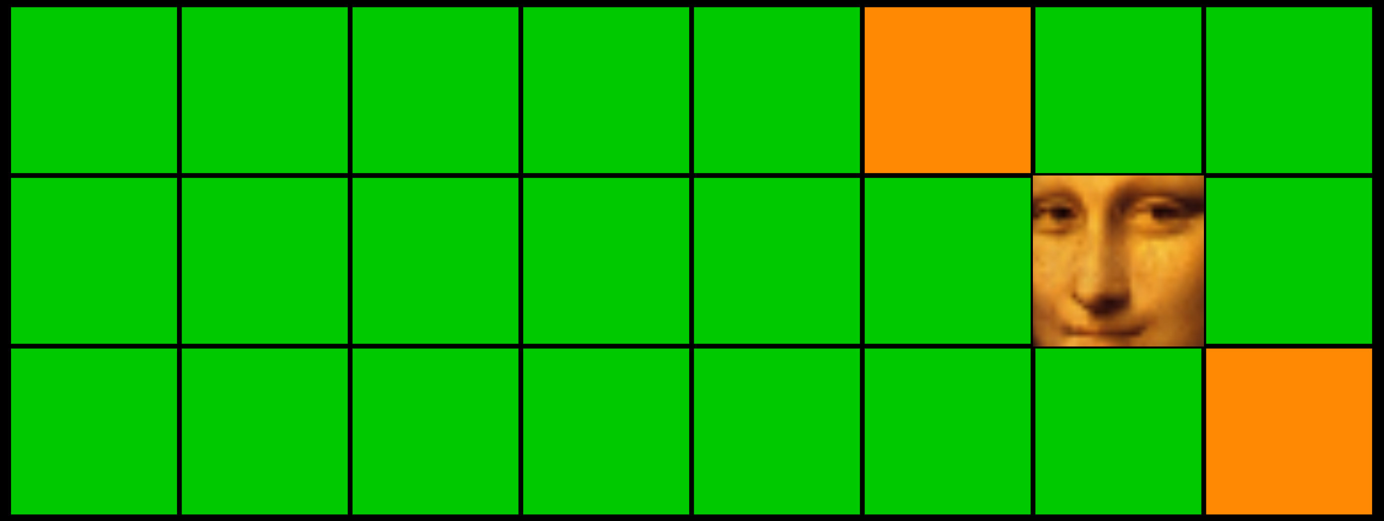


Deduplication implementation: KVM on Linux (KSM)

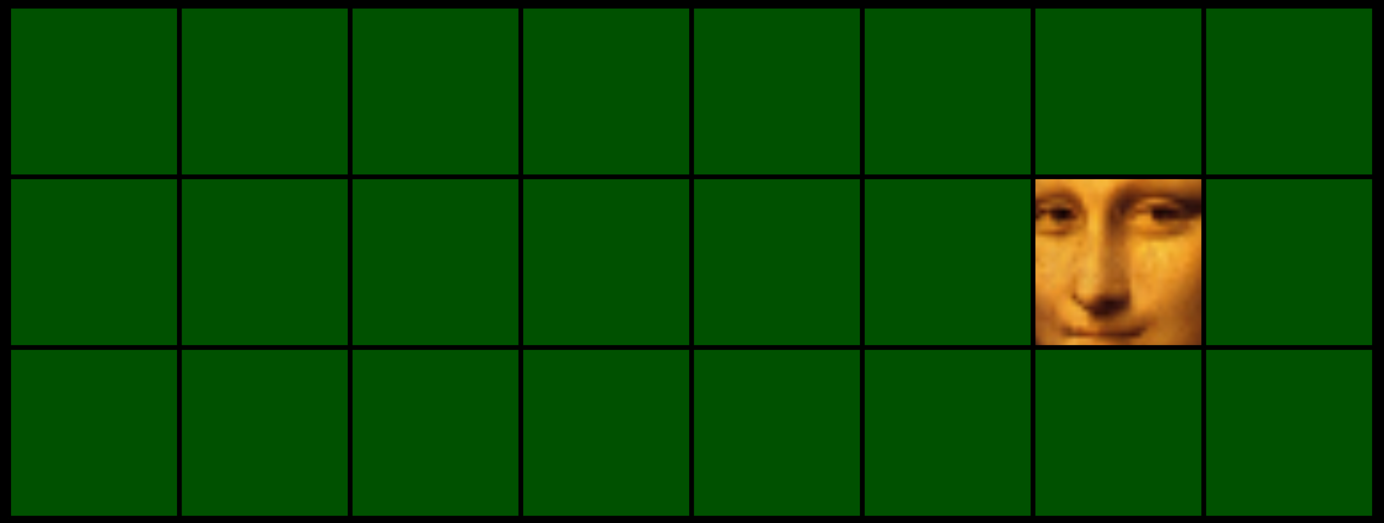
physical memory



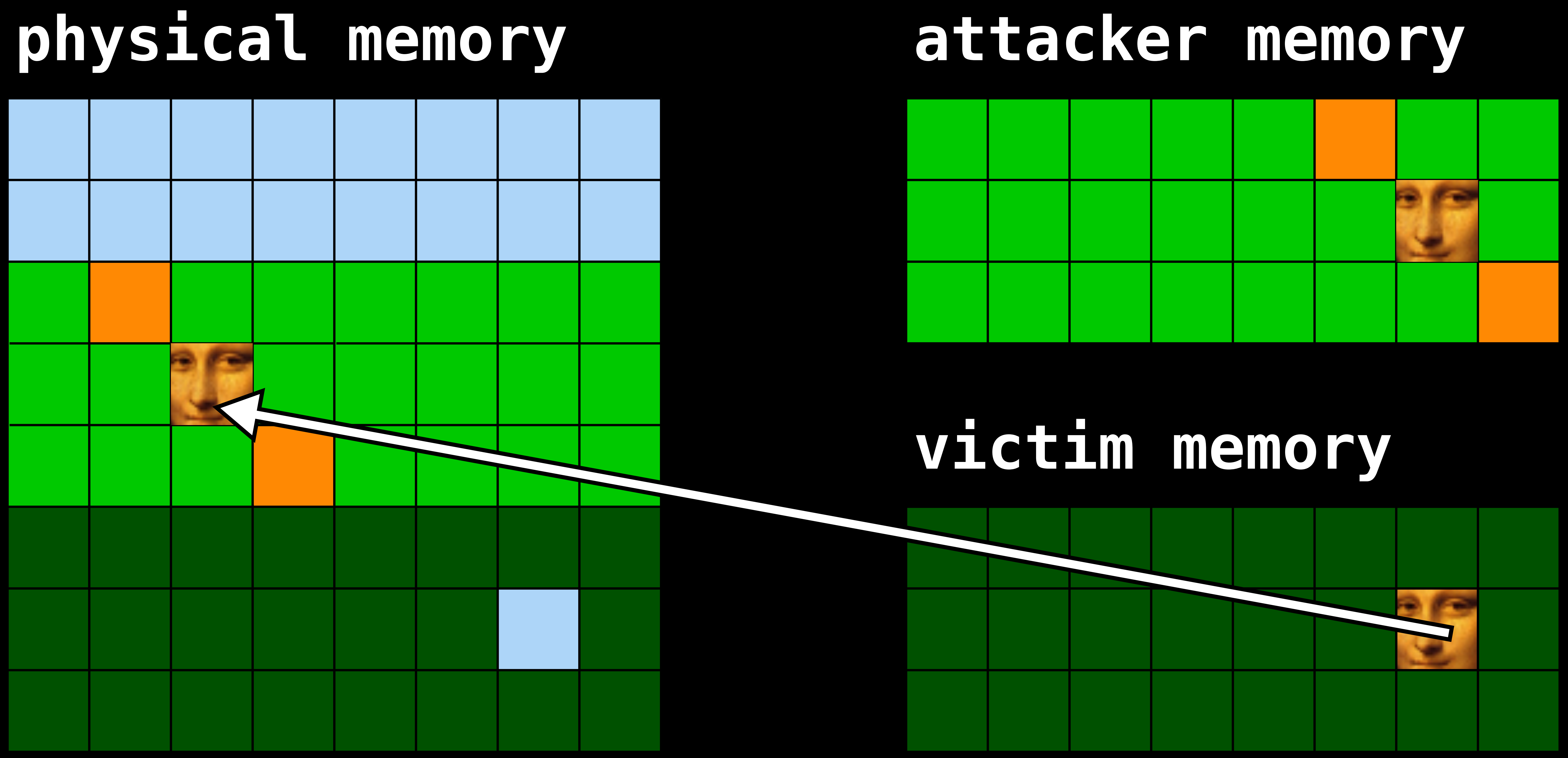
attacker memory



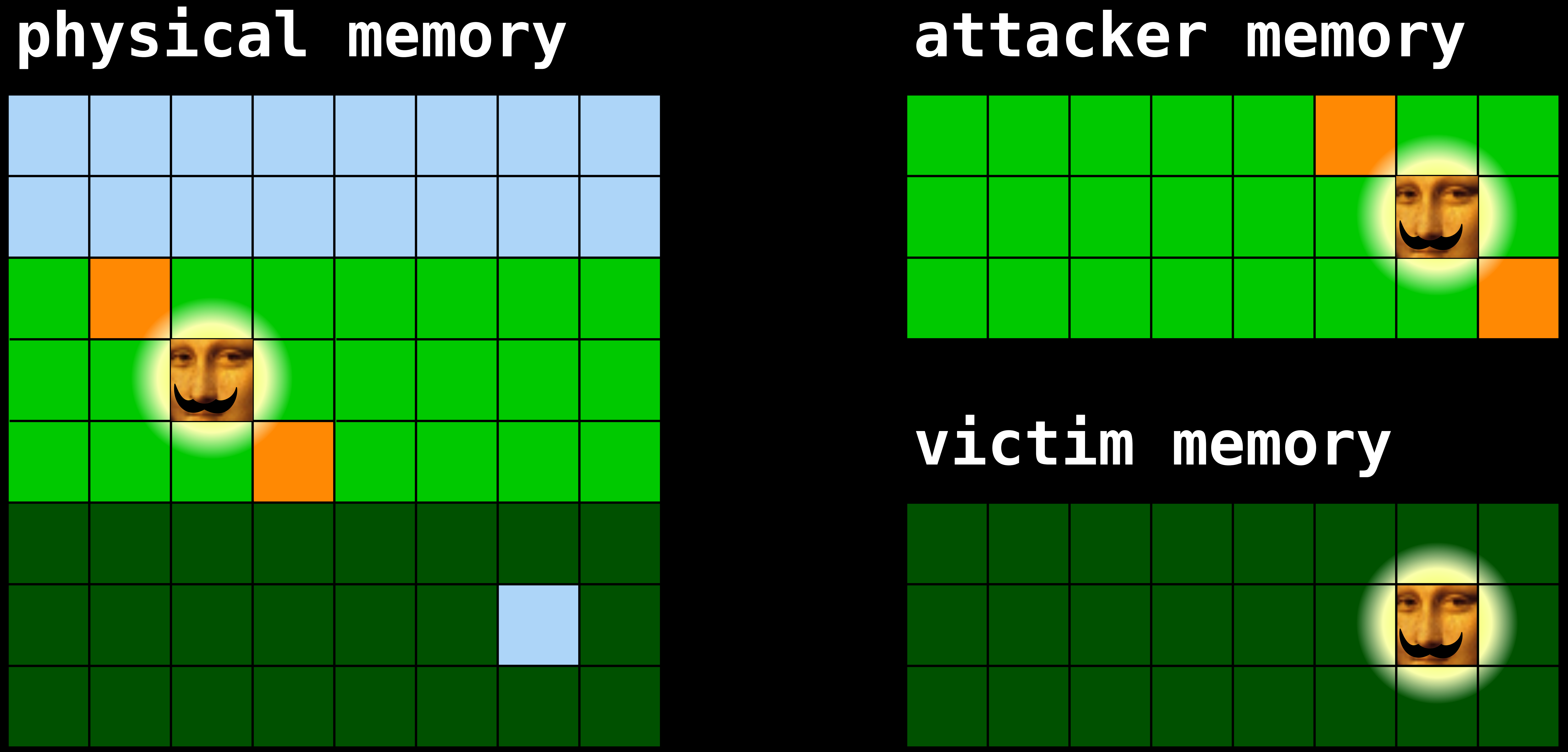
victim memory



Deduplication implementation:
KVM on Linux (KSM)



Deduplication implementation: KVM on Linux (KSM)



Example 1: OpenSSH

Target: ~/.ssh/authorized_keys

OpenSSH ~/.ssh/authorized_keys

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC52/Uk84iUmmic
el7ESr+/D/PWZ6Ljkhlu8yv35bEEoTwXm9eGxJyzV+1s68tRyzpD
3VQvwSHiKqDnCg+OtaAo0KvCqZcoBQFB9XawIfJI5dSeGtcUBuok
Uv+TlmAZ+D9MNNAxjuSBBH0ShbaiH65imlauISfR3VZWFE7uy6sB
26j52LhWG5BRwSkMnMRN2E2fqHaP96J9ROFlHuykw8jwUXJwl4kJ
8vRo1uhX0SVu8Z9wGrKR5b+GQWJ3Ph7vjoMVU/KoAbWnNnYKR8IT
BnkPD0LrEyAKRygEfi7gwcixOvQR79by8lL6ypJ4kM5eyobSBsNC
jmghxQj8RRzGUtd1 victim@laptop
```

Exponent   Modulus (p * q)

OpenSSH ~/.ssh/authorized_keys

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC52/Uk84iUmmic
el7ESr+/D/PWZ6Ljkhlu8yv35bEEoTwXm9eGxJyzV+1s68tRyzpD
3VQvwSHiKqDnCg+OtaAo0KvCqZcoBQFB9XawIfJI5dSeGtcUBuok
Uv+TlmAZ+D9MNNAxjuSBBH0ShbaiH65imlauISfR3VZWFE7uy6sB
26j52LhWG5BRwSkMnMRN2E2fqHaP96J9ROFlHuykw8jwUXJwl4kJ
8vRo1uhX0SVu8Z9wGrKR5b+GQWJ3Ph7vjoMVU/KoAbWnNnYKR8IT
BnkPD0LrEyAKRygEfi7gwcixOvQR79by8lL6ypJ4kM5eyobSBsNC
jmghxQj8RRzGUtd1 victim@laptop
```

Exponent

Modulus (p' * q' * r' ...)

Example 1: OpenSSH

Target: ~/.ssh/authorized_keys

> Flip a bit in the RSA modulus

> Factorize it

> Reconstruct the new private key

Example 2: GPG & apt-get

Targets: sources.list

flip package repository domain name

eg. ubuntu.com -> ubunvu.com

Example 2: GPG & apt-get

Targets: sources.list

+

GPG keyring

corrupt signing key

Conclusion

Conclusion

> Memory deduplication is dangerous

Conclusion

> Memory deduplication is dangerous

> Be aware of the security implications

Conclusion

> Memory deduplication is dangerous

> Be aware of the security implications

> Well, or just disable it

Erik Bosman

erik@minemu.org

@brainsmoke

Antonio Barresi

antonio.barresi@xorlab.com

@AntonioHBarresi

Rowhammer (seaborn attack)