Hand Me Your SECRET, MCU!

Microarchitectural Timing Attacks on Microcontrollers are Practical

Cristiano Rodrigues | Sandro Pinto, PhD

(Centro ALGORITMI / LASI, Universidade do Minho)

Hand Me Your SECRET, MCU!

Microarchitectural Timing Attacks on Microcontrollers are Practical

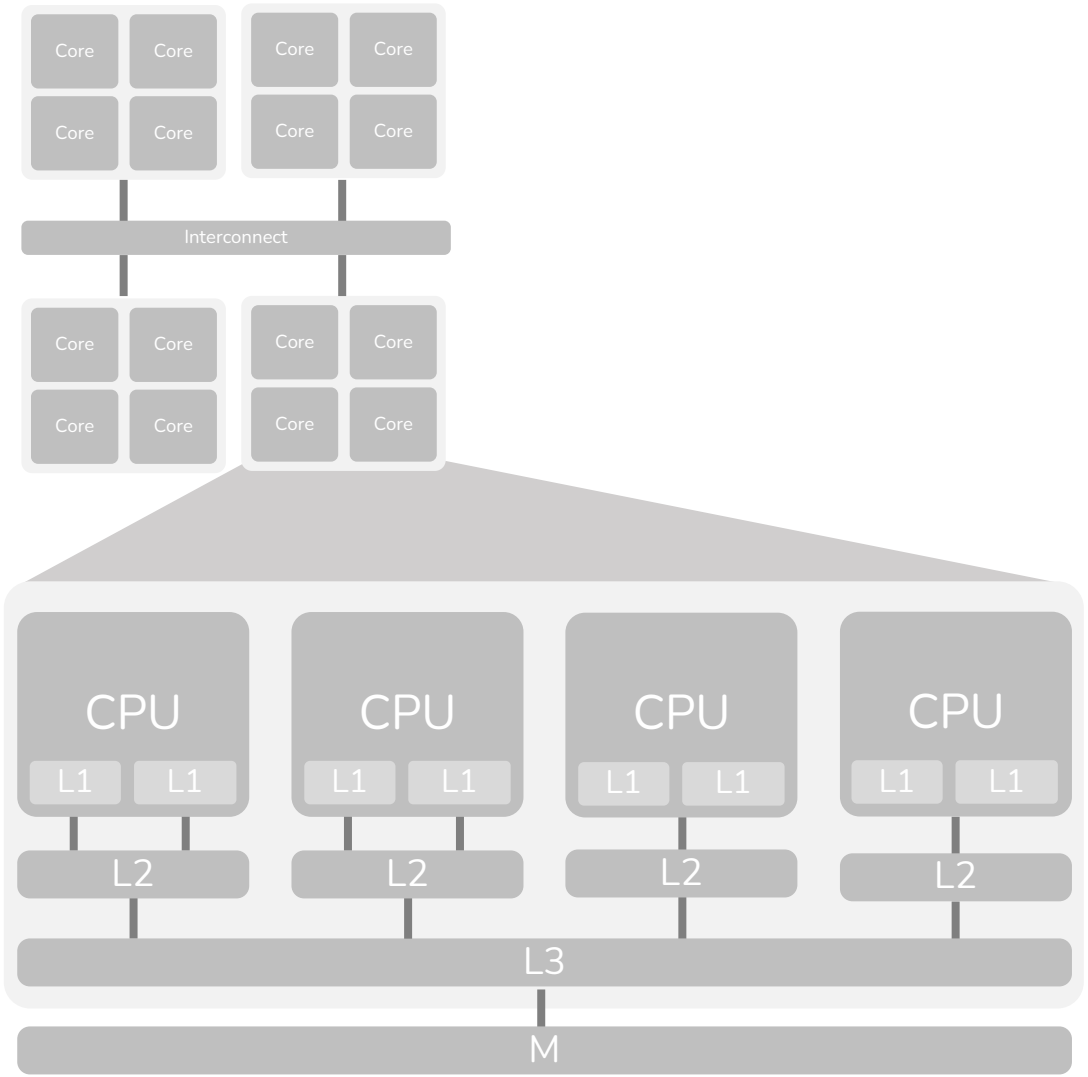
Cristiano Rodrigues | Sandro Pinto, PhD

(Centro ALGORITMI / LASI, Universidade do Minho)

CENTROALGORITMI

Microarchitectural

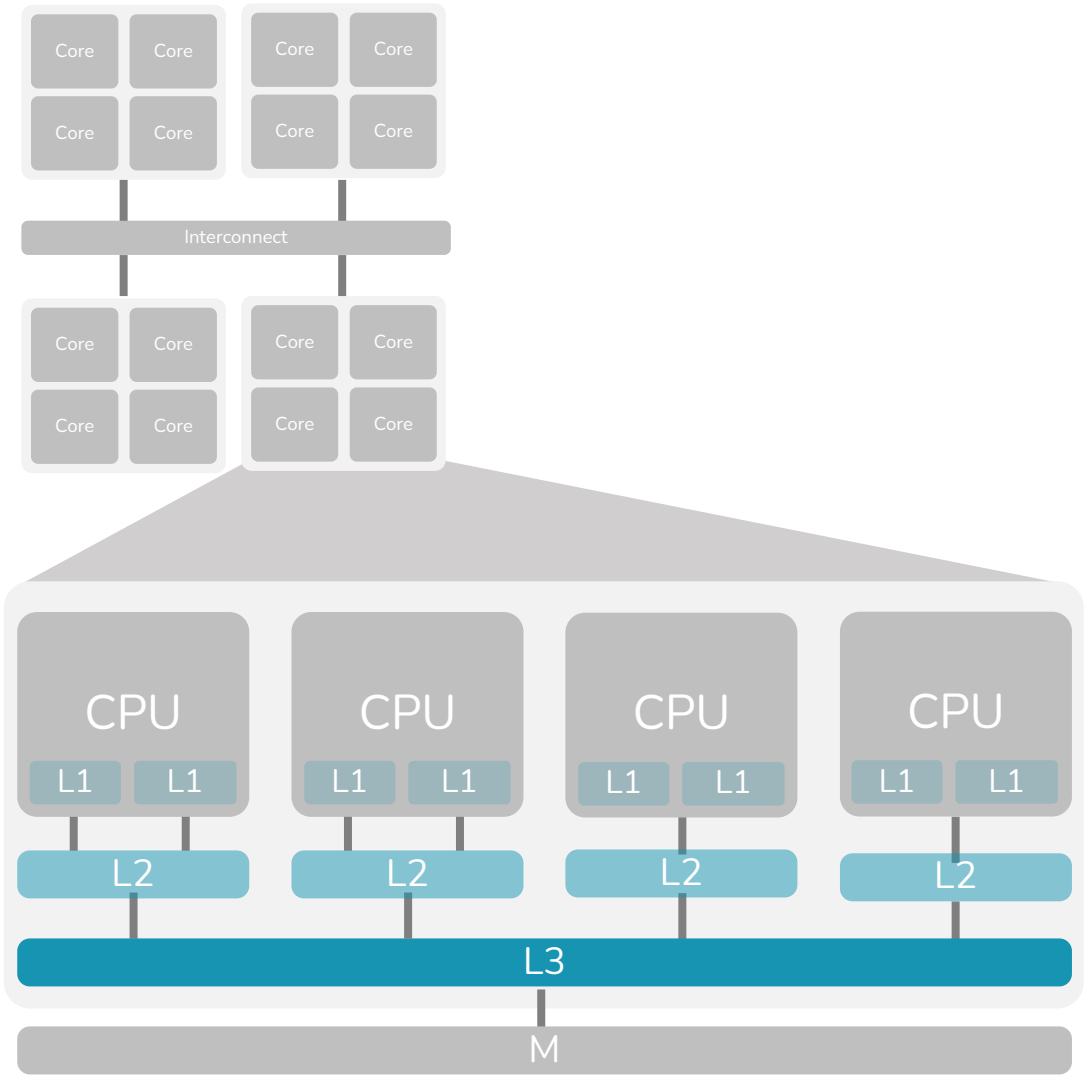
SIDE-CHANNEL Attacks



Microarchitectural SIDE-CHANNELS

Servers, PCs, Mobile

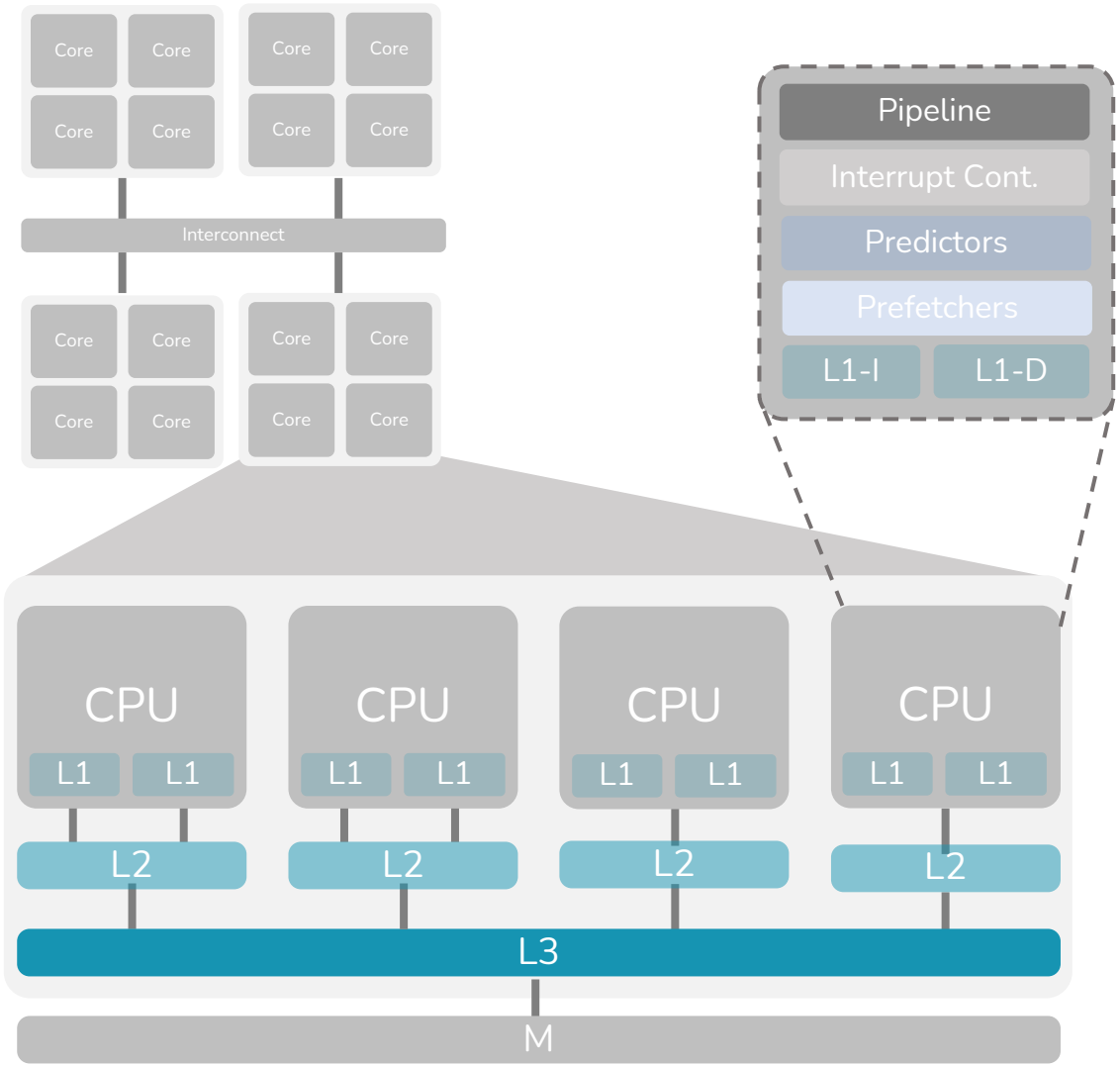
intel AMD arm



Microarchitectural SIDE-CHANNELS

Servers, PCs, Mobile

intel AMD arm

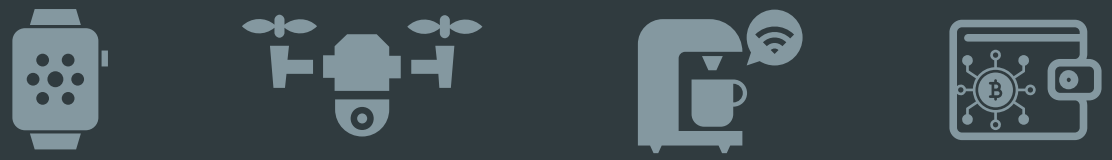


Microarchitectural SIDE-CHANNELS

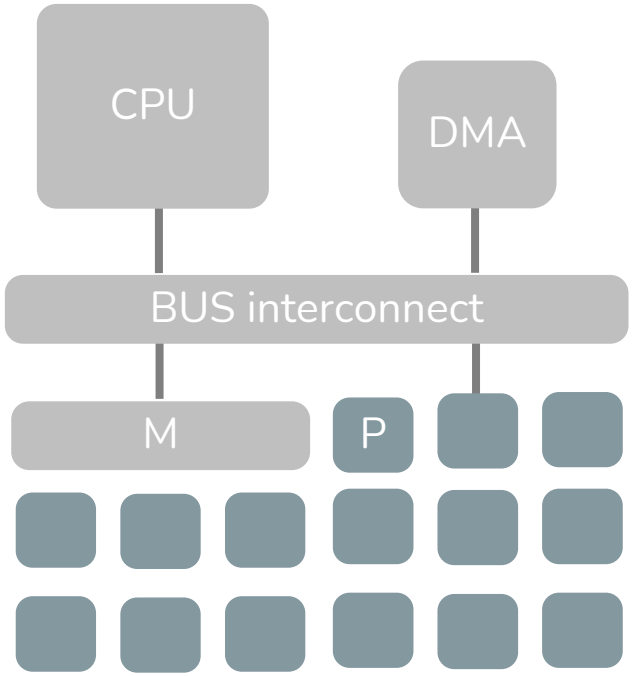
Servers, PCs, Mobile

intel AMD arm

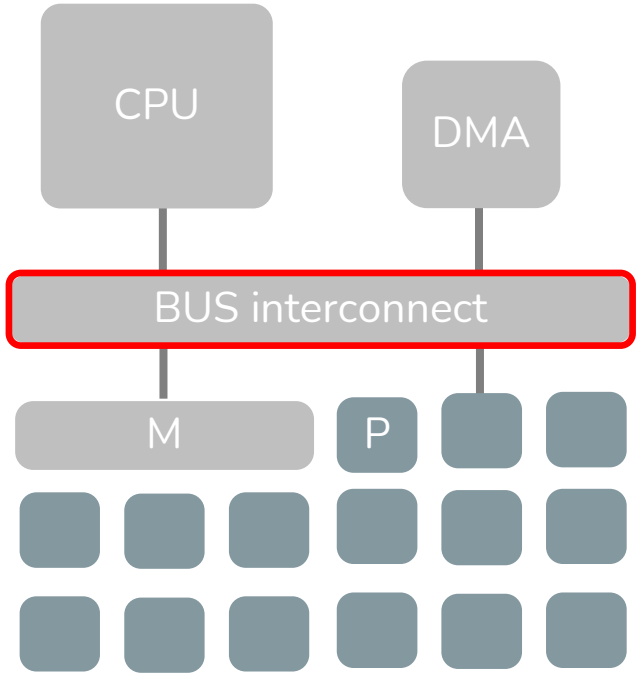
Microcontrollers



arm

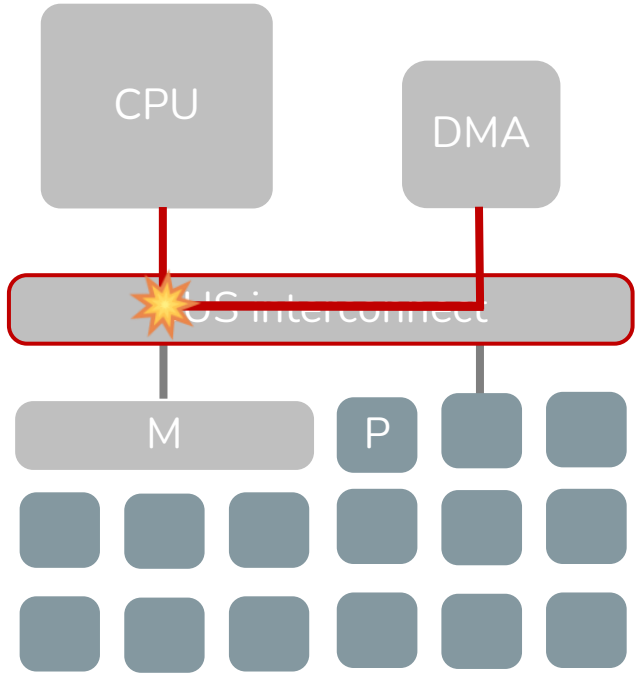


There is a common belief that MCUs are not vulnerable to microarchitectural timing side-channel attacks because their microarchitecture is intrinsically simple



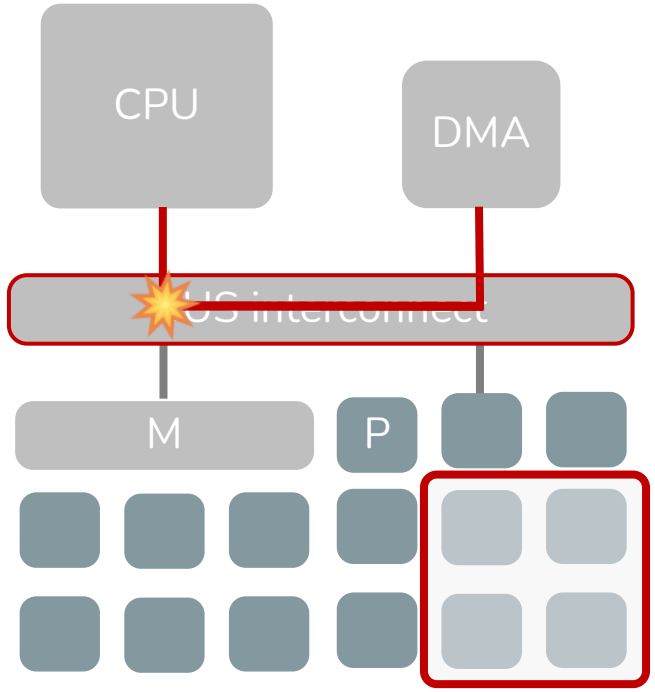
Novel Channel

BUS Interconnect



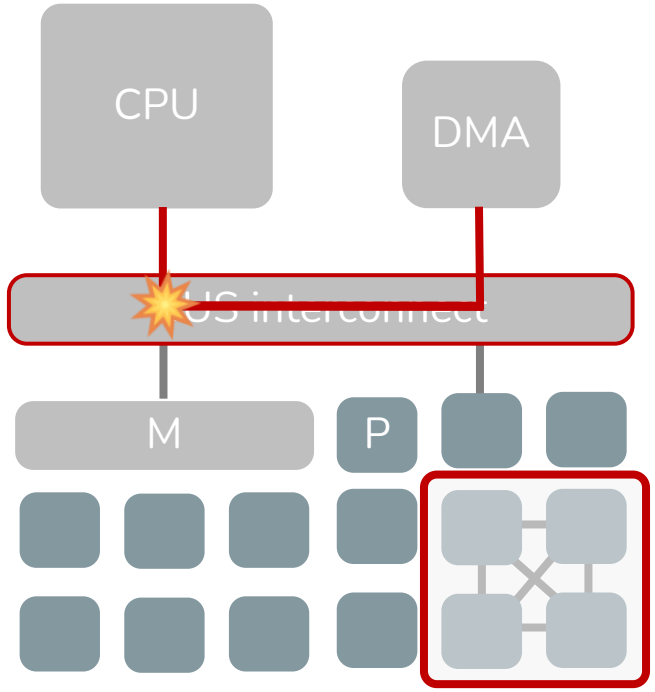
BUS Interconnect

Arbitration logic



Hardware Gadgets

Novel Concept

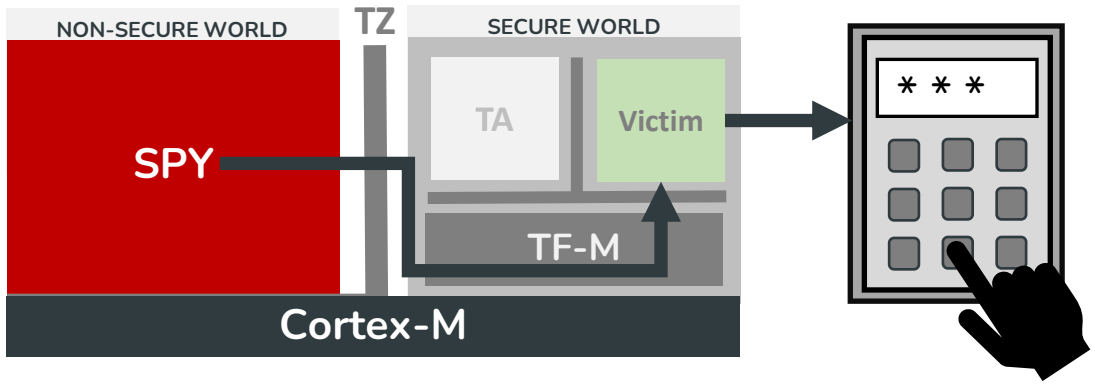


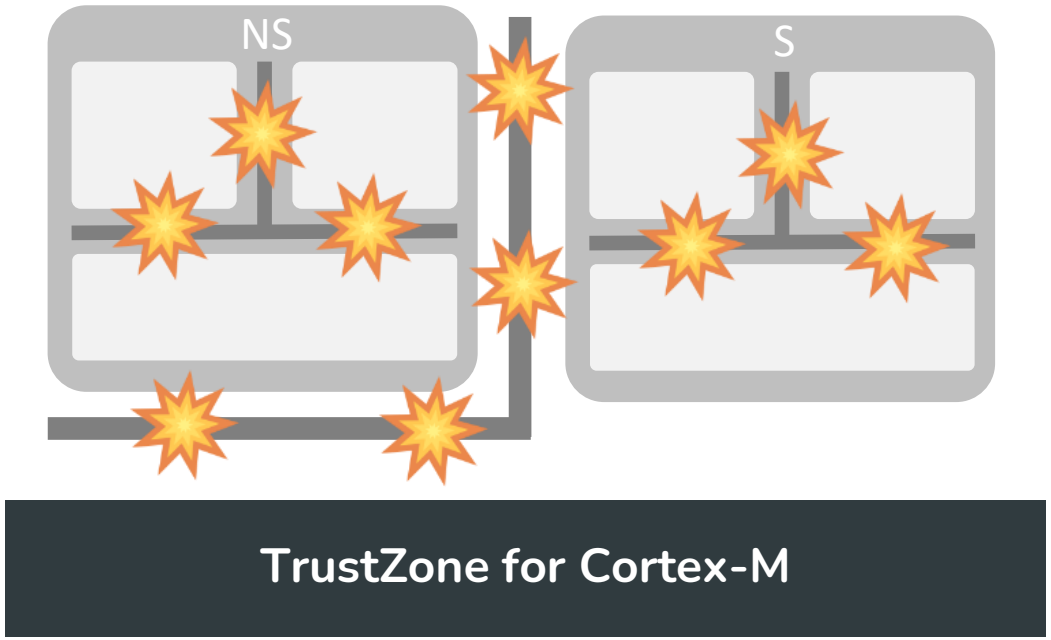
Hardware Gadgets

Smart Gadget Network

BUSted

Attack





BUSted

Attack

AGENDA

01 **Introduction**
Motivation and Side-Channels “101”

02 **Hidden Threat**
Novel Side-Channel Source: MCU Bus Interconnect

03 **“Toy” Example**
Basic Attack Example

04 **BUSted Attack**
Microarchitectural Side-Channel Attacks on MCUs

05 **“Live” Demo**
Demo of BUSted Attack

06 **Summary**
Responsible Disclosure and BH Sound Bytes

Introduction

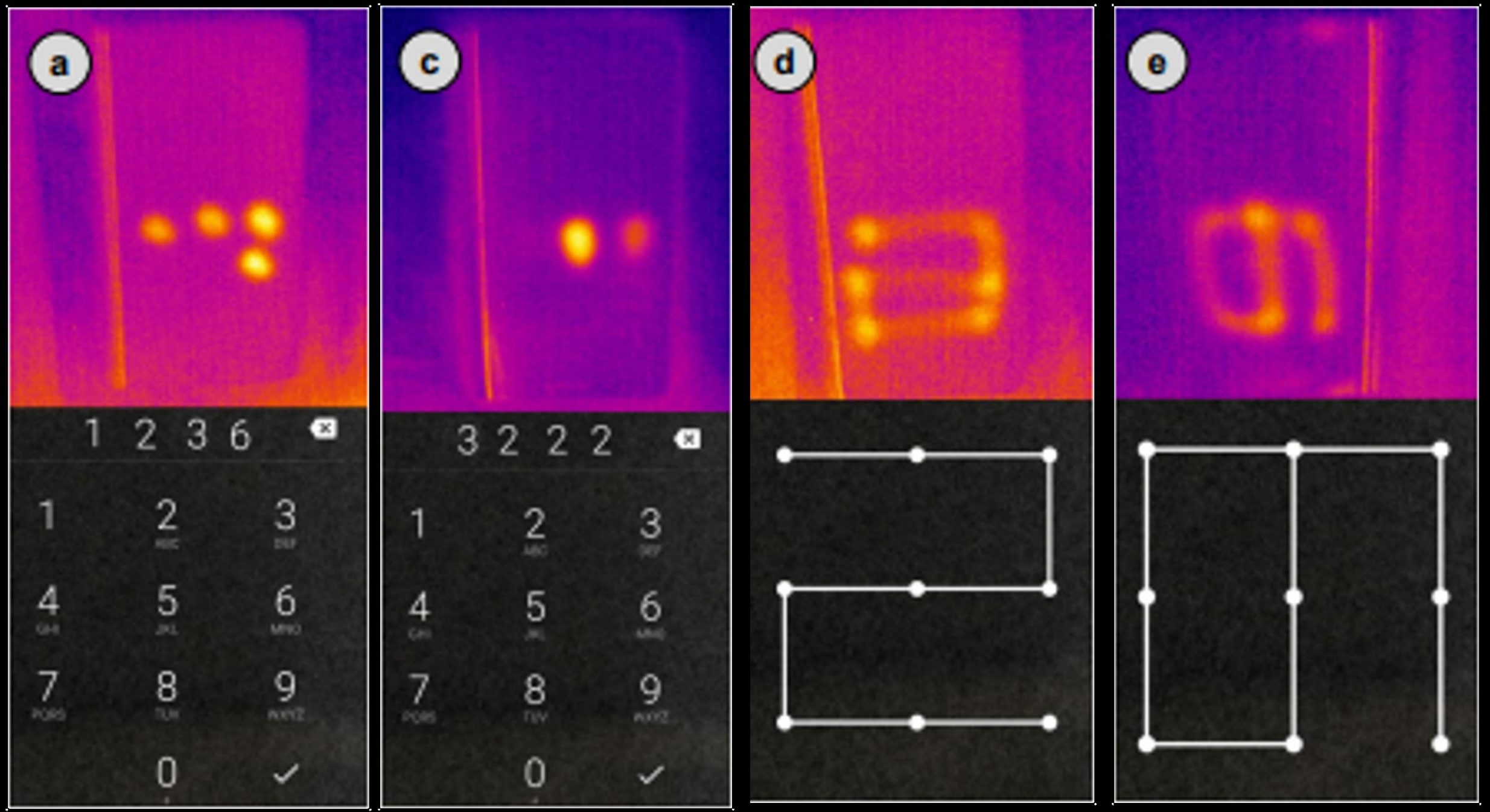
Motivation and Side-Channels “101”

HEAT / POWER

SOUND

ELECTROMAGNETIC

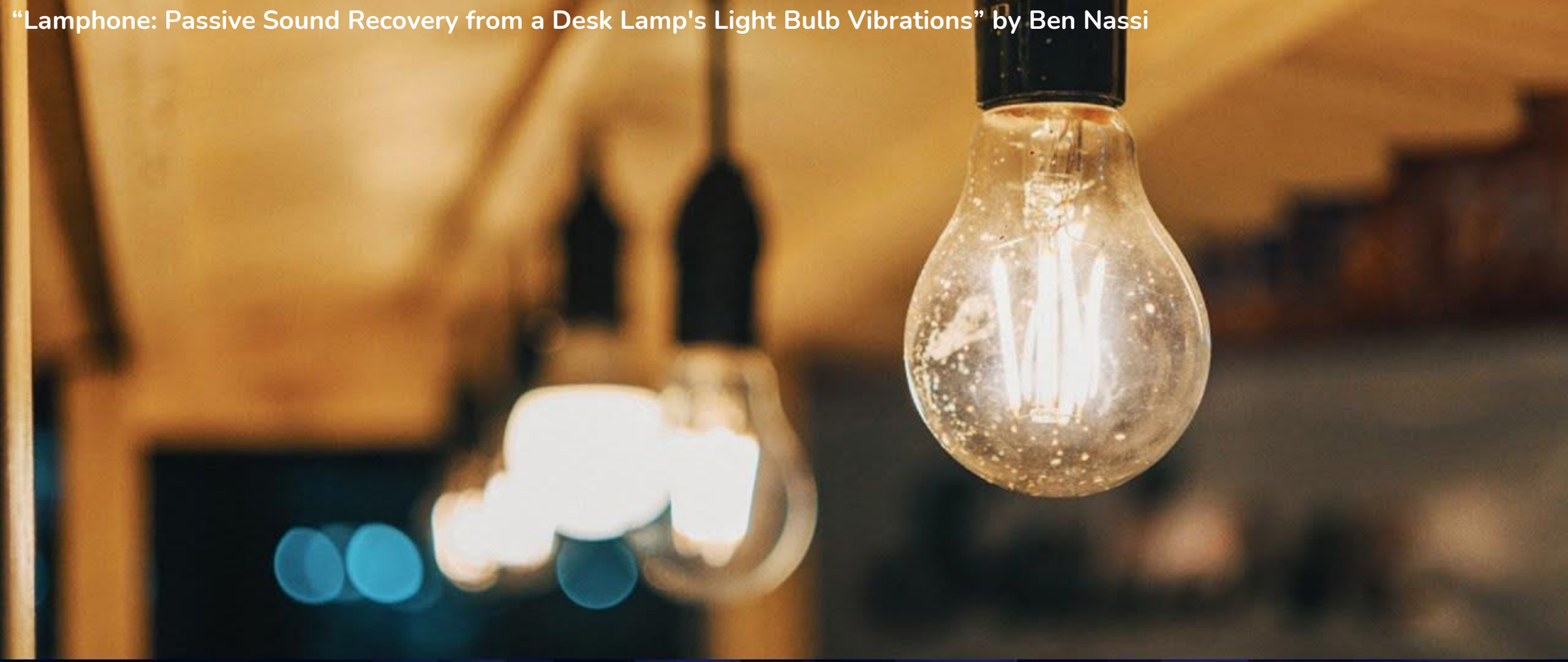
TIME



“Stay Cool! Understanding Thermal Attacks on Mobile-based User Authentication” by Abdelrahman and Khamis

“RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis” by Daniel Genkin

“Lamphone: Passive Sound Recovery from a Desk Lamp's Light Bulb Vibrations” by Ben Nassi



Lamphone Lightbulb Spies

MELTDOWN

SPECTRE


TECH / SECURITY
Protecting your
latest security
Spectre is goi
Dieter Bohn
Spectre attacks against
Google
COMPUTER SCIENCES
Editors' notes
still a
vulnerability found
Intel CPUs vulnerable
attack
By Bill
Researchers discover new side-channel attack that
works against Intel CPUs
The Swiss cheese of Chipzilla has more holes than we thought
By Adrian Potoroaca April 25, 2023 at 10:13 AM
The Hacker News
New Cache Side Channel
Targeted Online Users
A team of academic researchers has warned of a novel cache-base
deanonymization attack that could be used to defeat anonymity...
Jul 15, 2022
MAY 21, 2018
CPUs
03:55 PM
channel attack vul
vulnerability to th
2023
attack can
tre, Another Scar
em
GX),
dogged Intel CPUs.
0
110
New sp
from Intel ch
Intel-specific vulnerability wa
3:31 PM GMT • January 26, 2018
ssTalk attack impacts Intel's
ile, desktop, and server CPUs

APUs

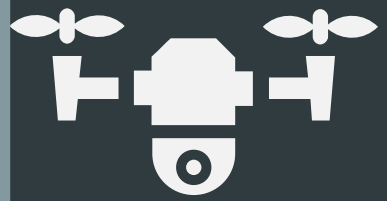
Servers

Mobile

PCs

Gateways

MCUs



Drones

Hardware Wallets

Appliances

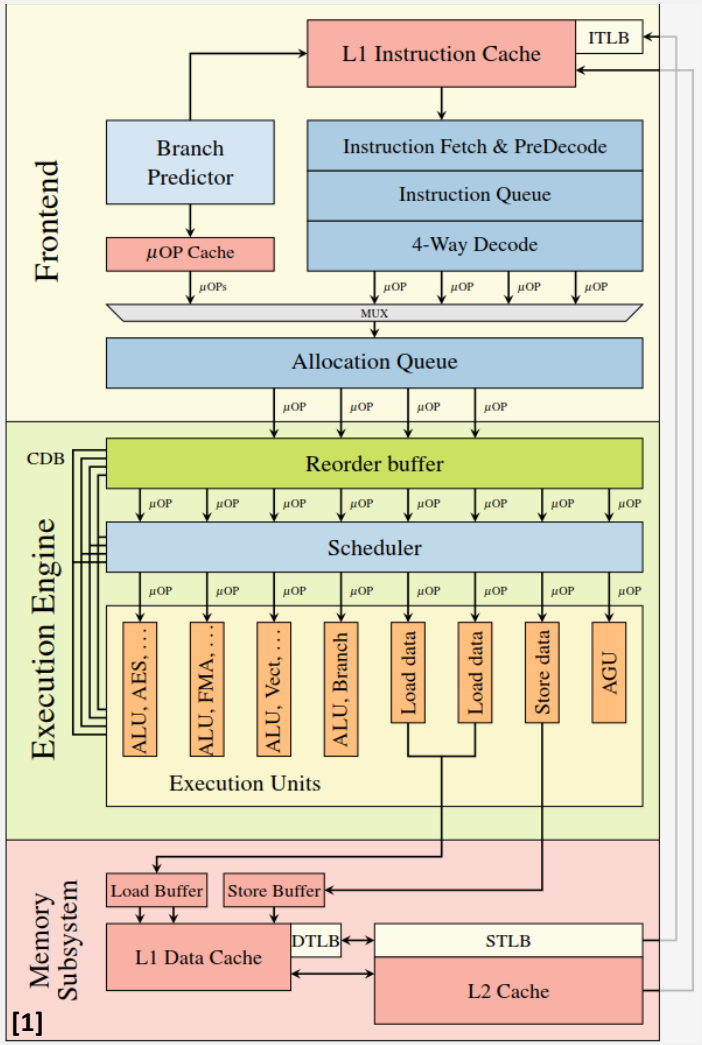
Wearables

Computing spectrum

Intel Skylake Microarchitecture

Arm Cortex-M33 Microarchitecture

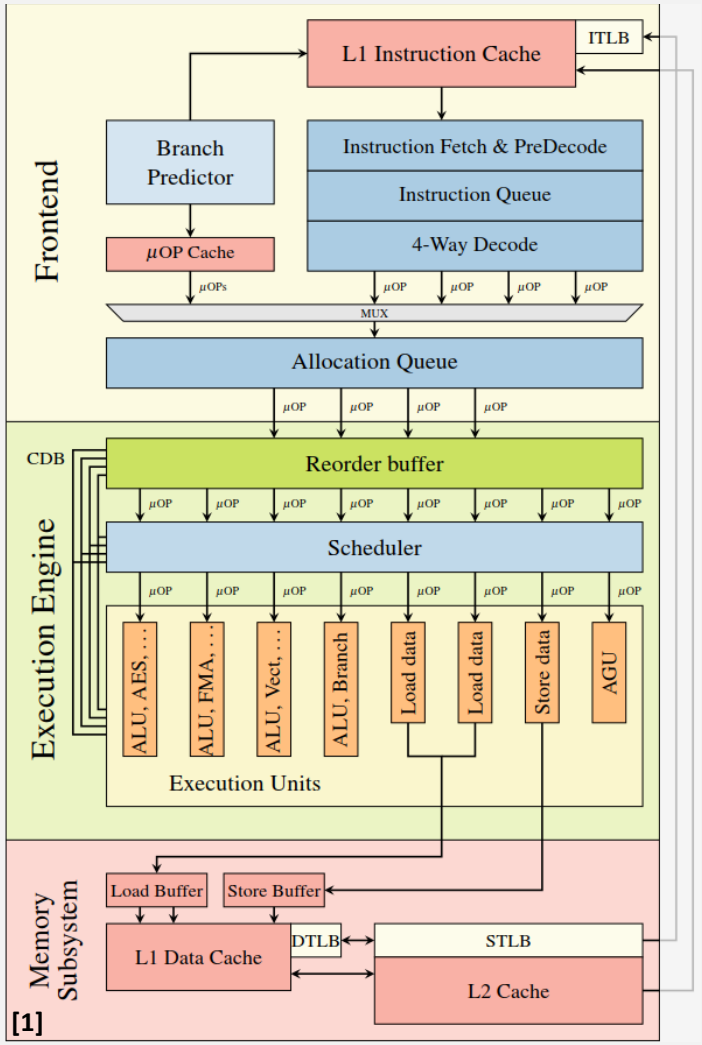
Intel Skylake Microarchitecture



[1] - Meltdown: Reading Kernel Memory from User Space

Arm Cortex-M33 Microarchitecture

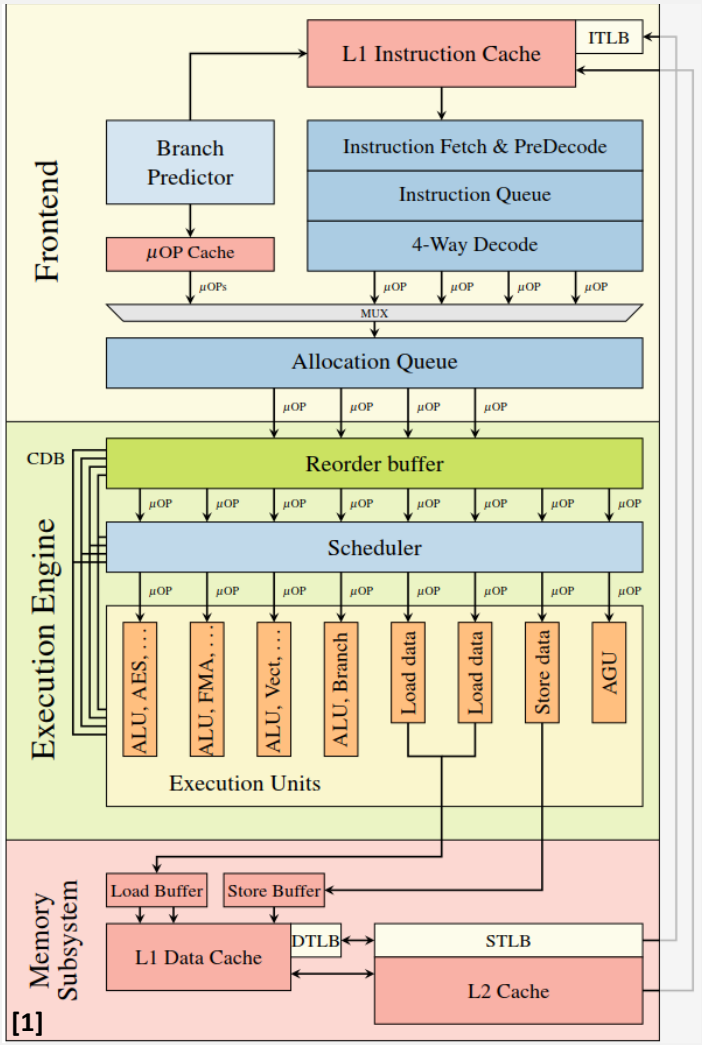
Intel Skylake Microarchitecture



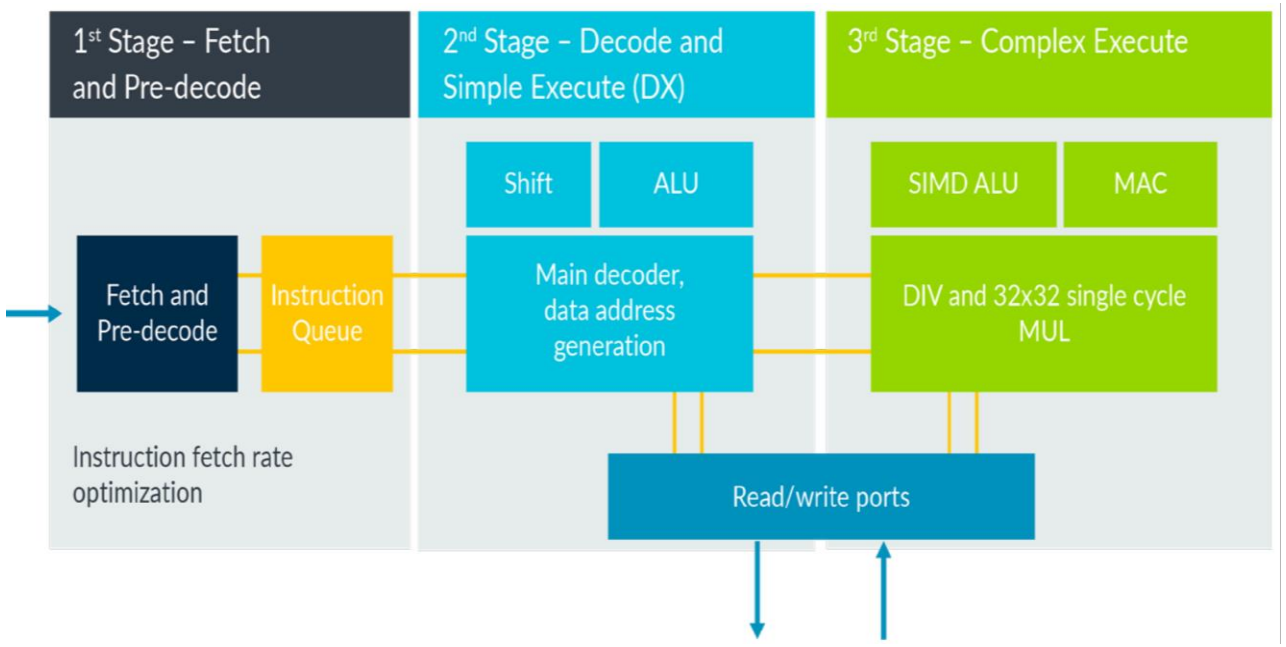
Arm Cortex-M33 Microarchitecture

[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture



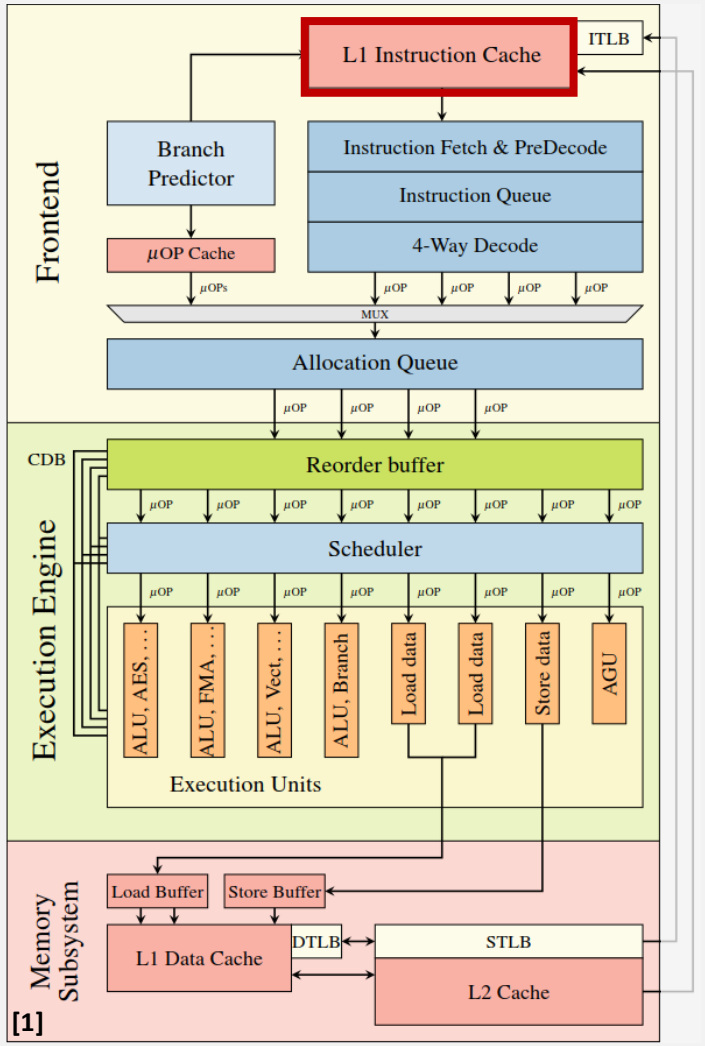
Arm Cortex-M33 Microarchitecture



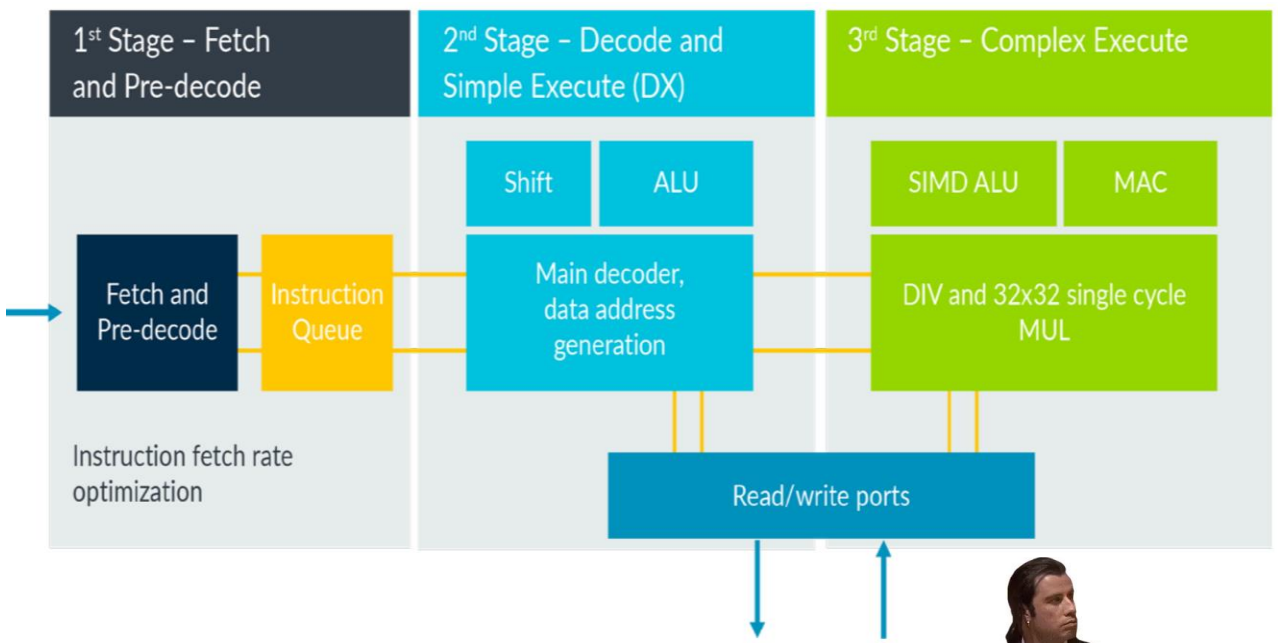
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I



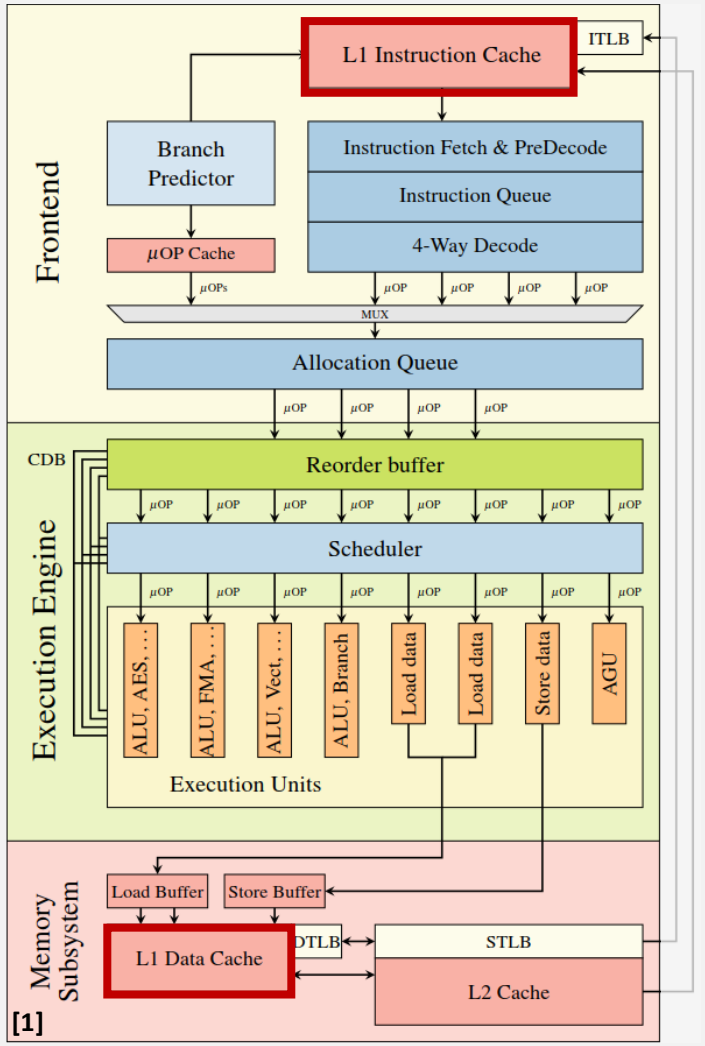
Arm Cortex-M33 Microarchitecture



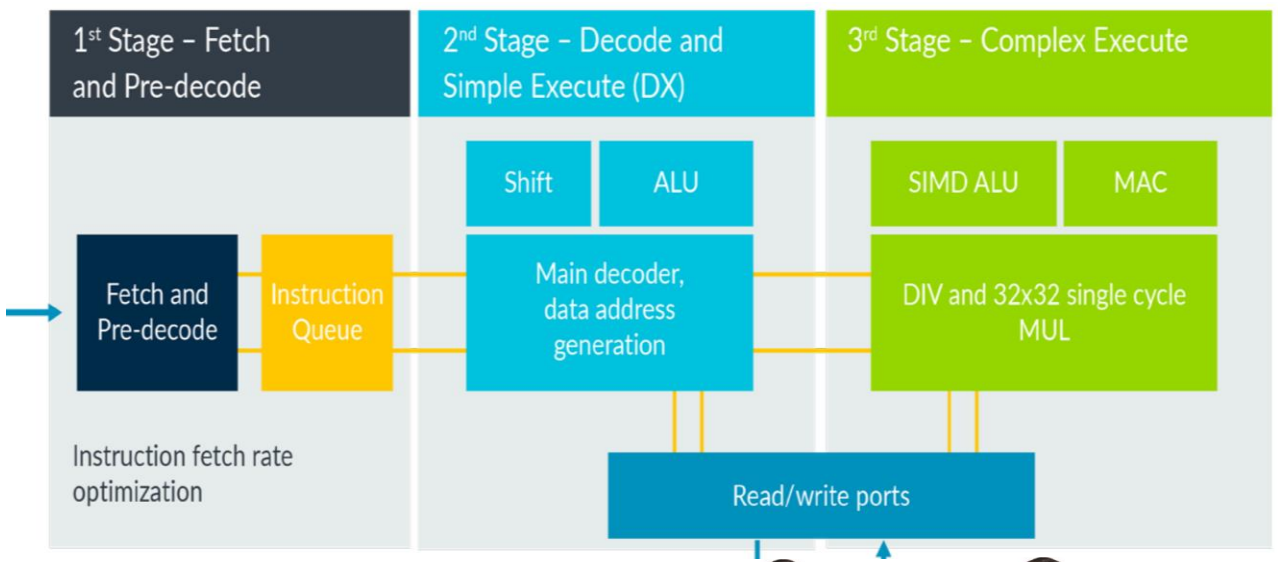
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D



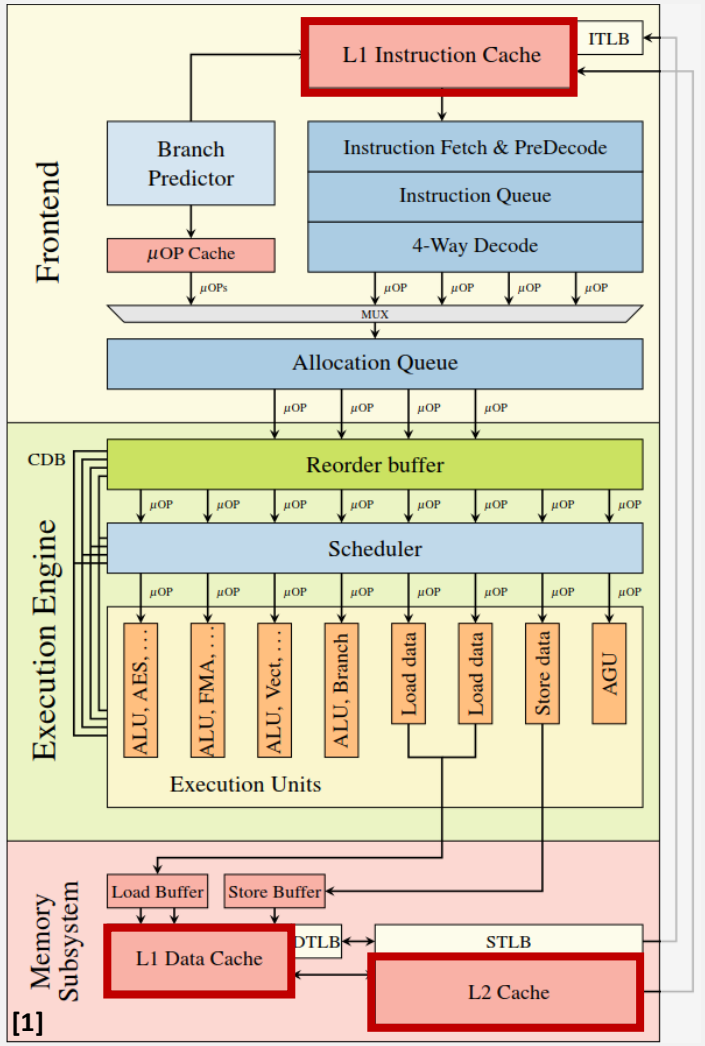
Arm Cortex-M33 Microarchitecture



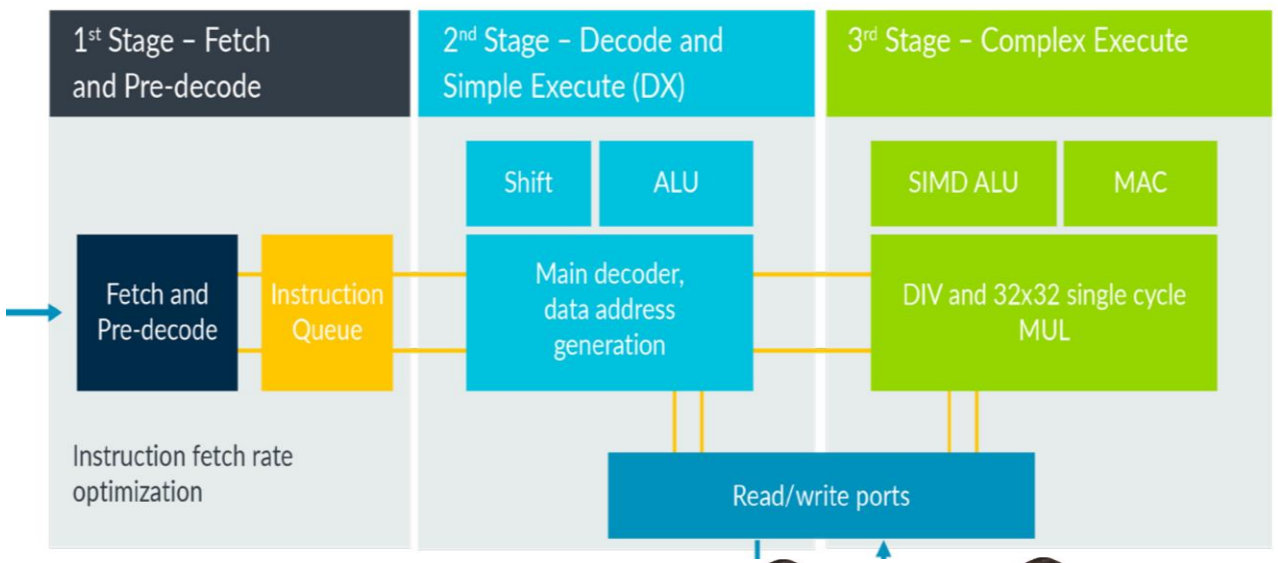
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2



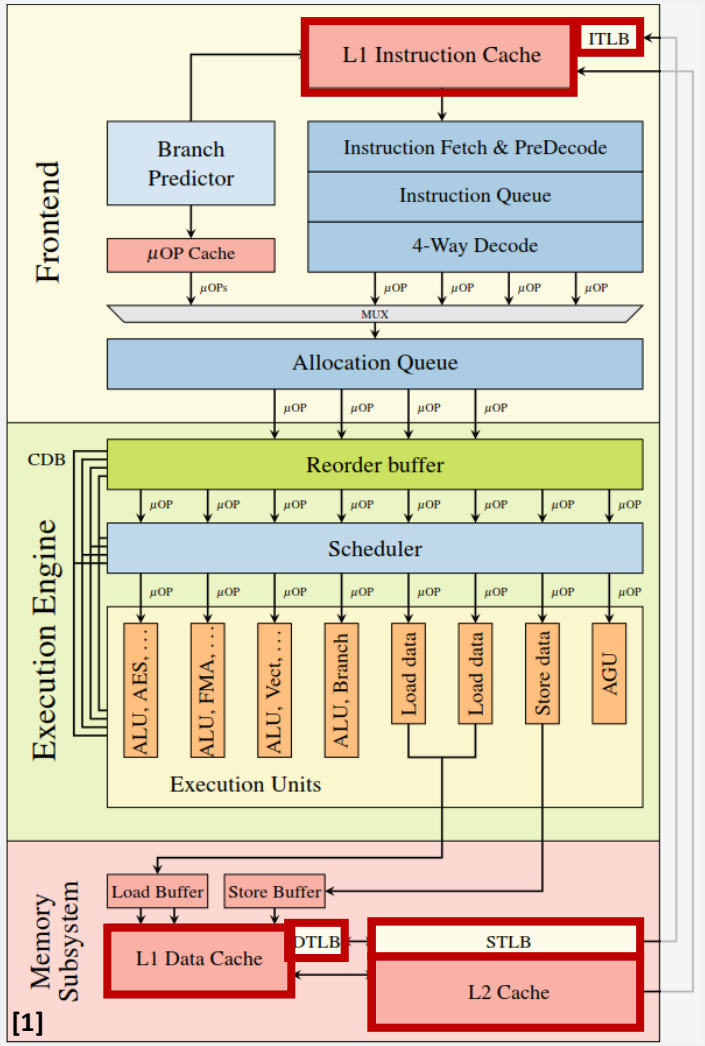
Arm Cortex-M33 Microarchitecture



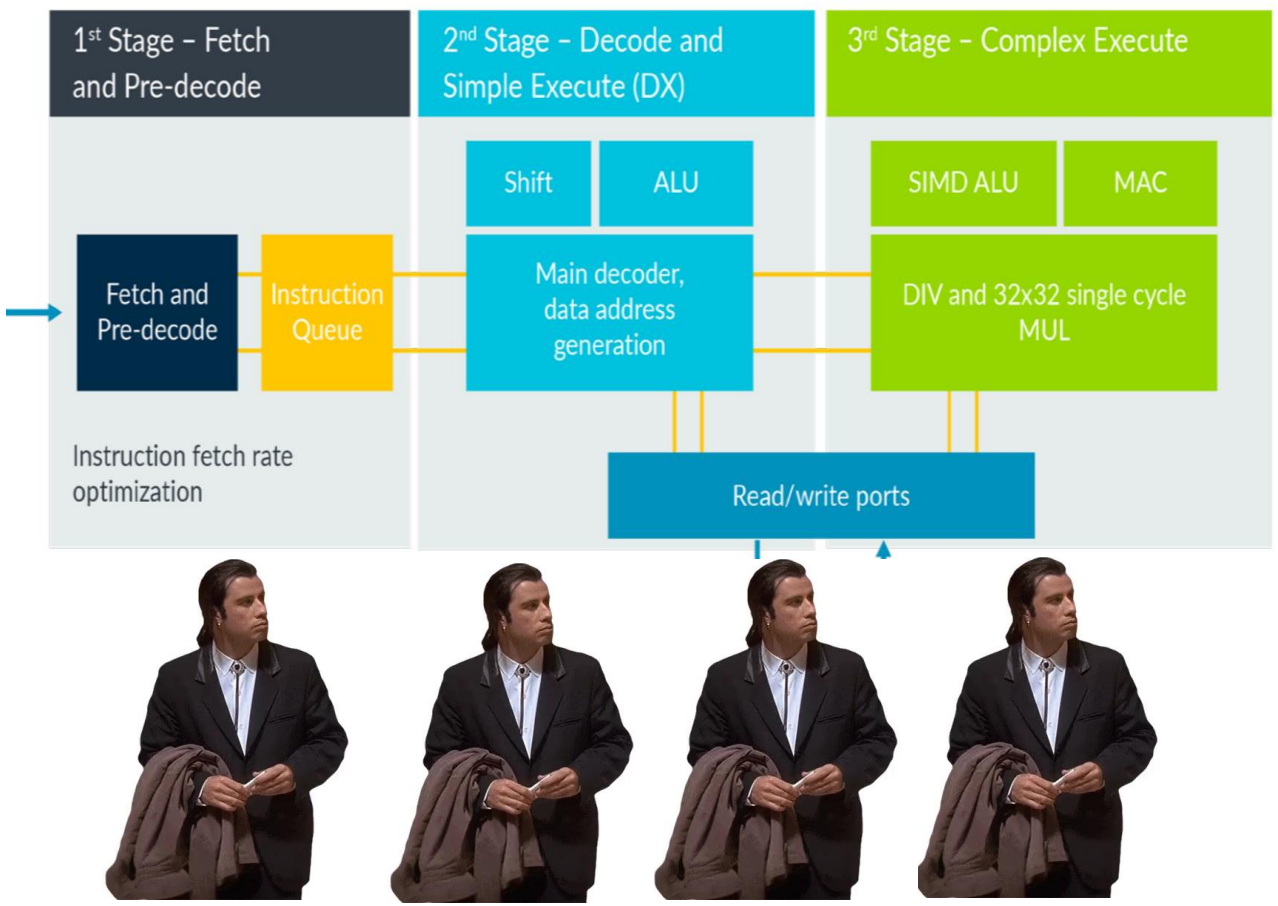
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2
- TLBs



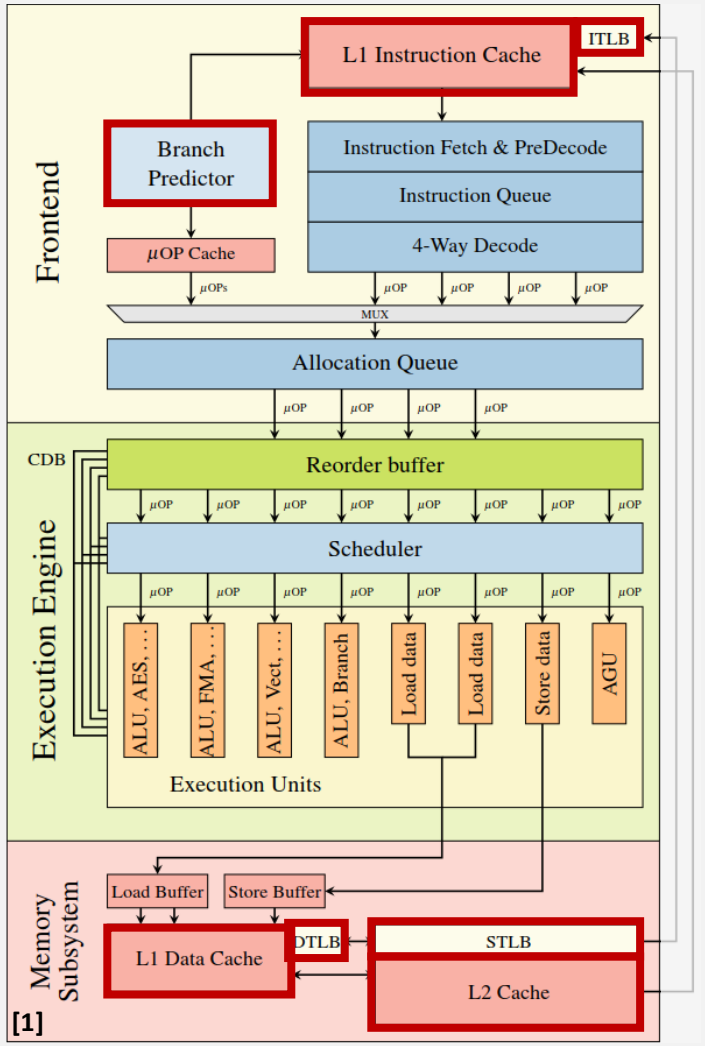
Arm Cortex-M33 Microarchitecture



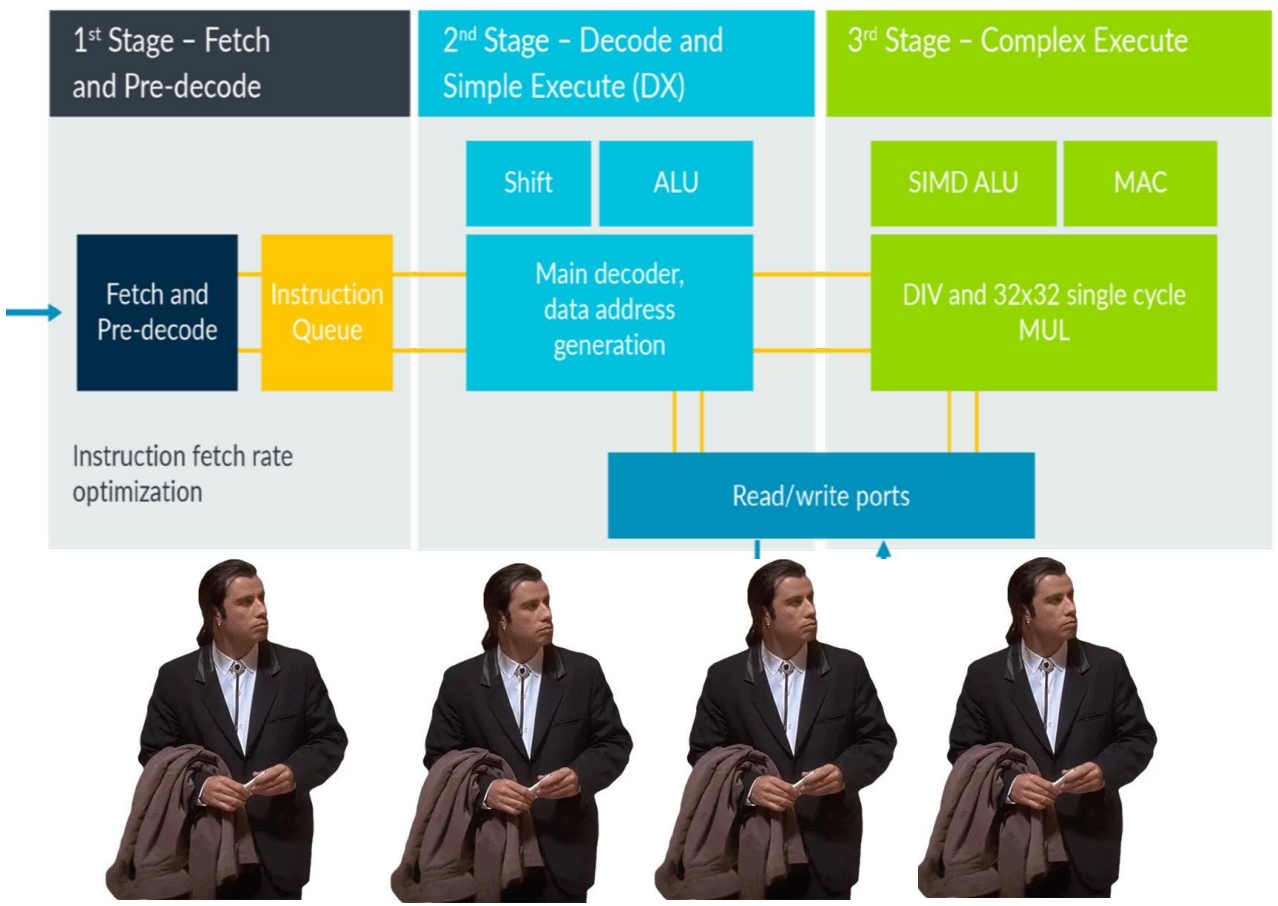
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2
- TLBs
- BP



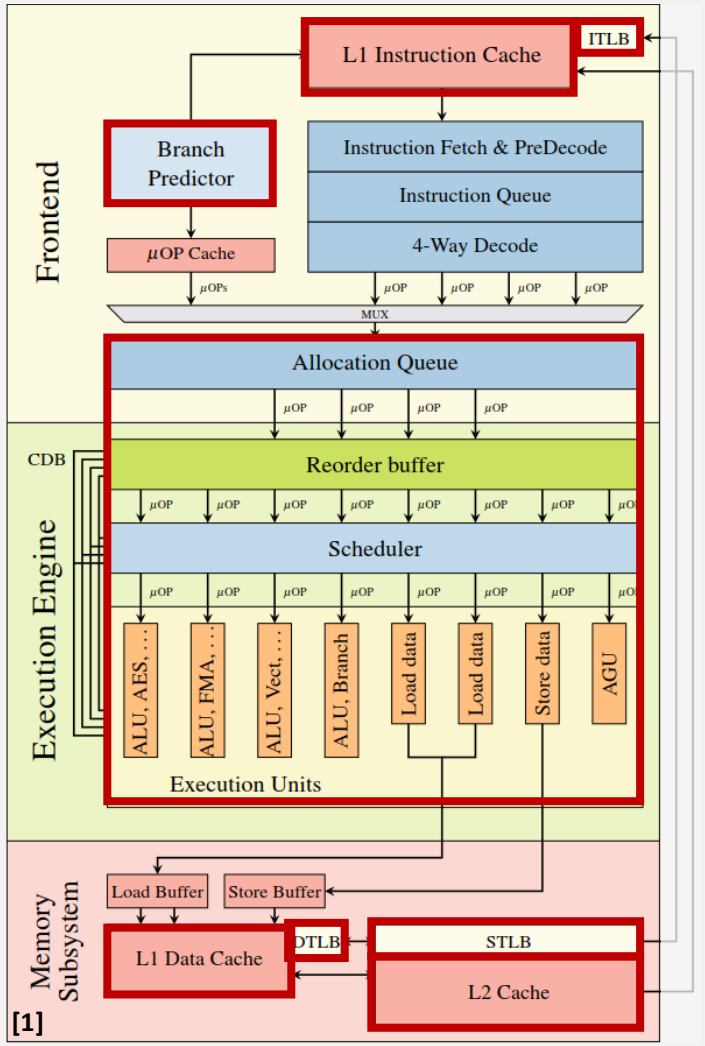
Arm Cortex-M33 Microarchitecture



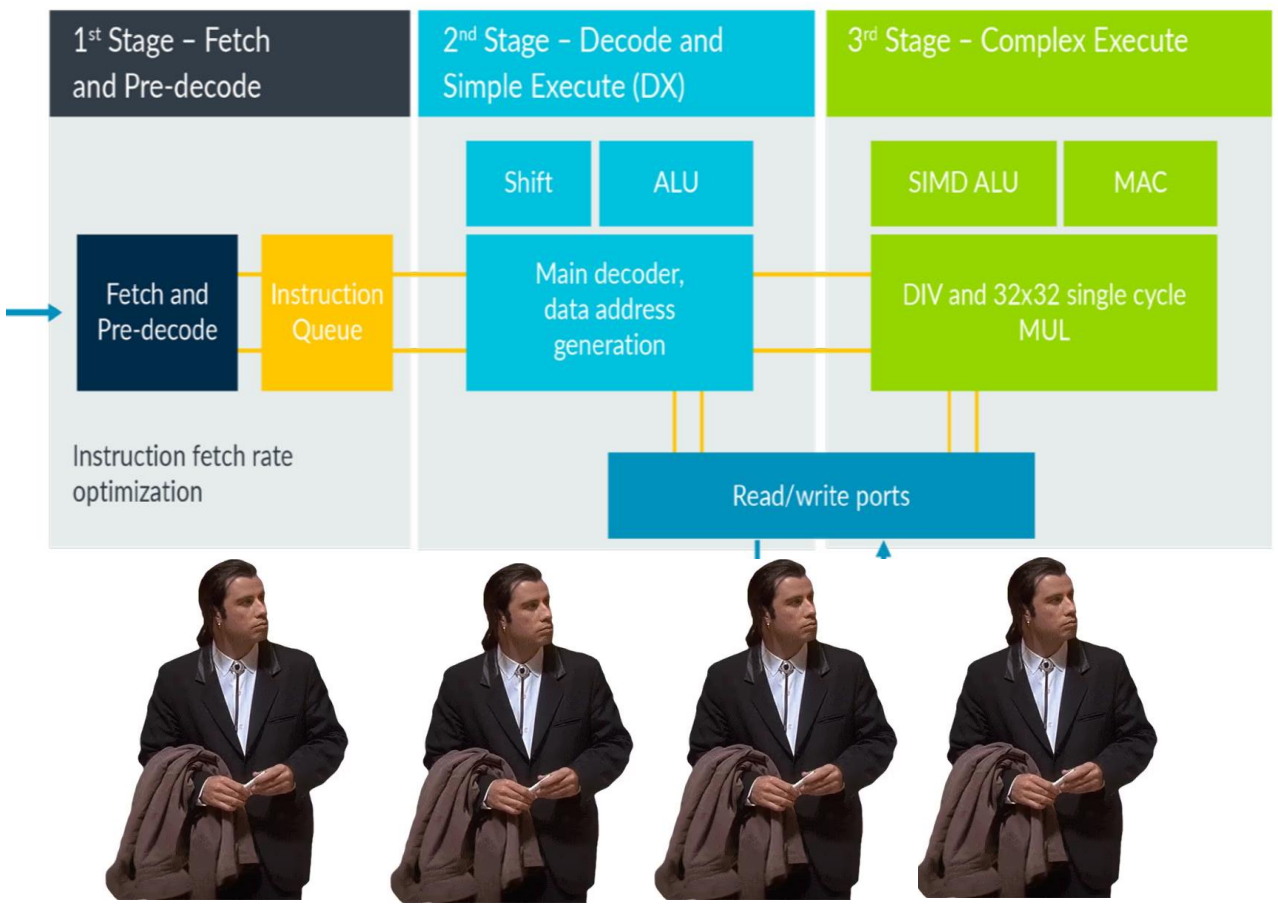
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2
- TLBs
- BP
- OoO



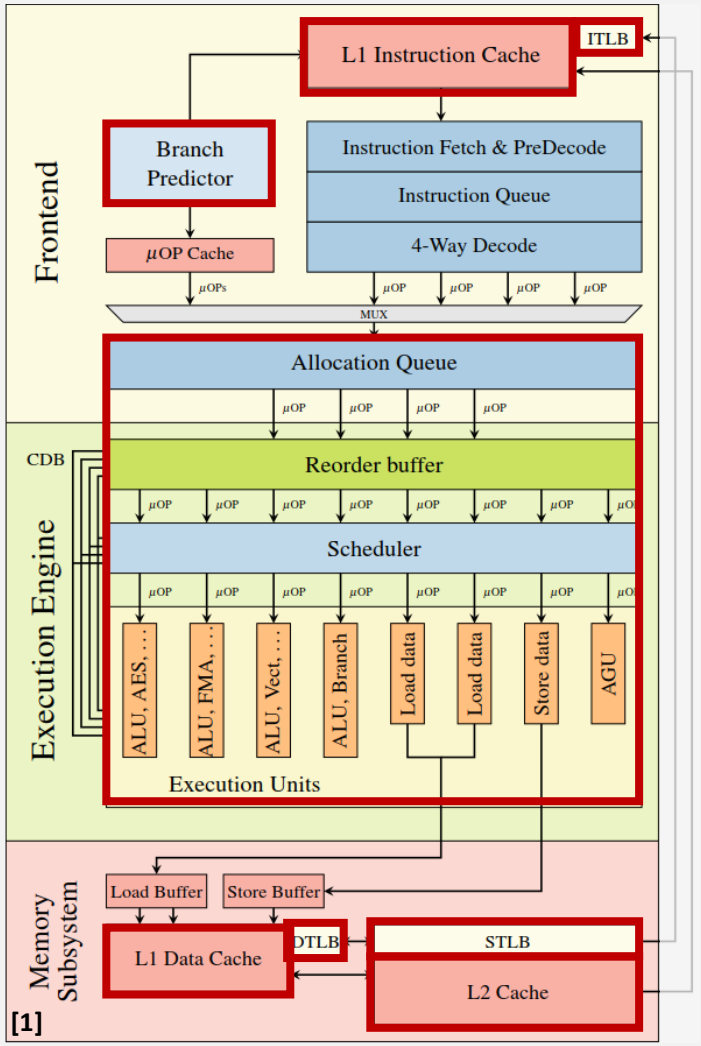
Arm Cortex-M33 Microarchitecture



[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

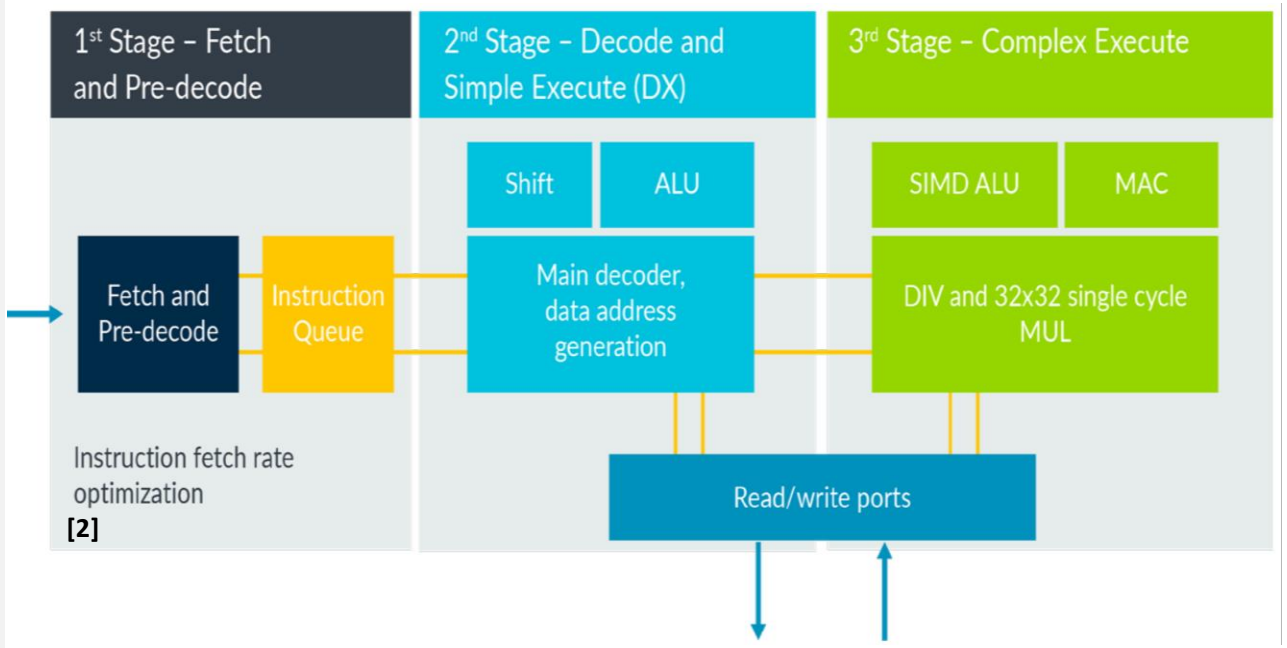
Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2
- TLBs
- BP
- OoO



[1]

Arm Cortex-M33 Microarchitecture

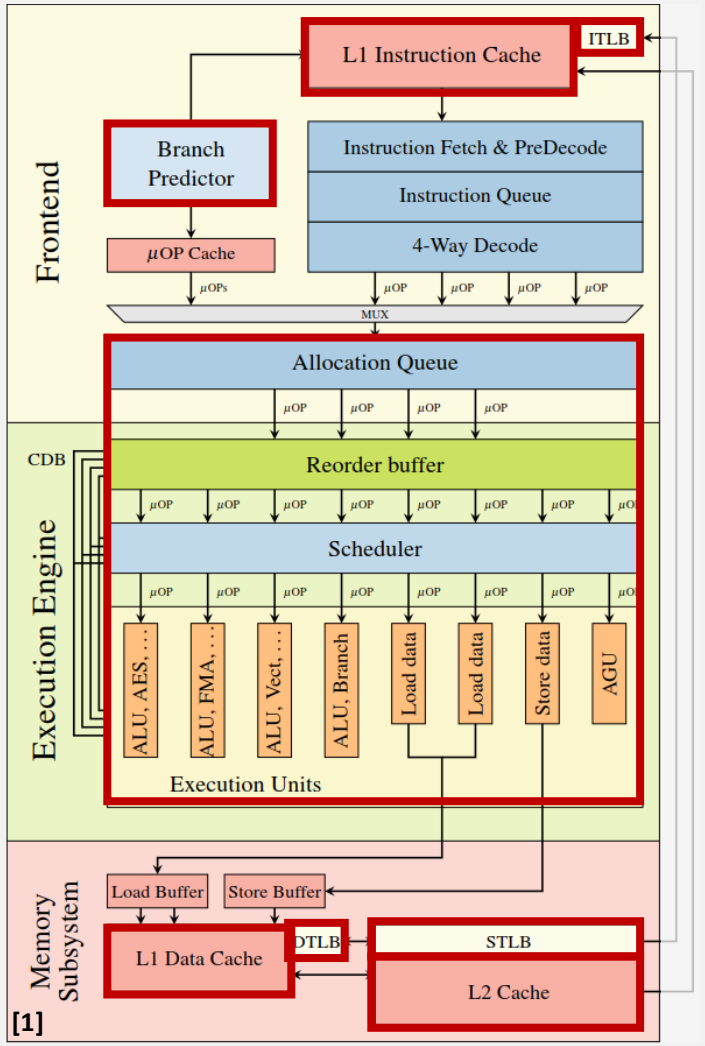


[2]

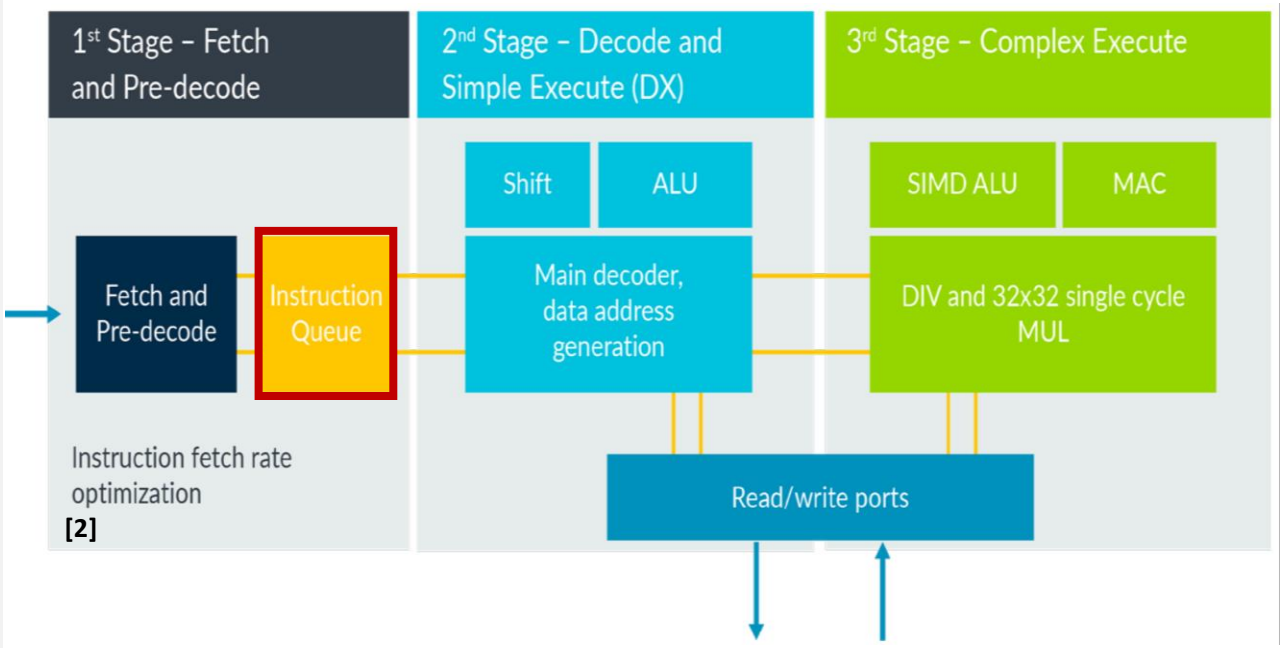
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2
- TLBs
- BP
- OoO



Arm Cortex-M33 Microarchitecture

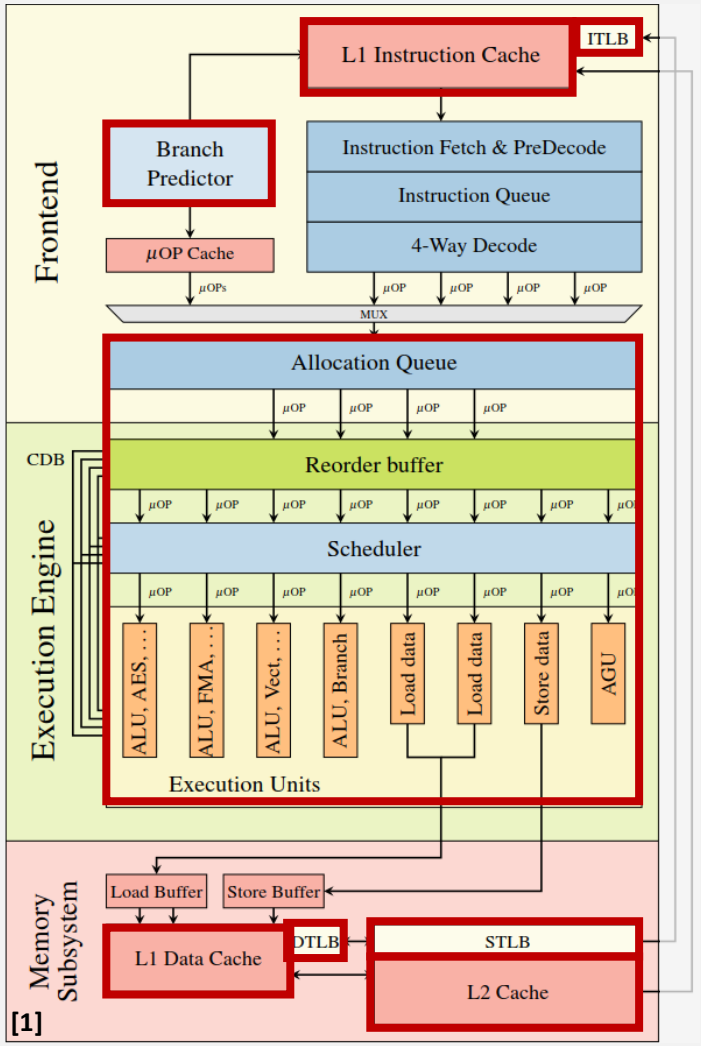


- Instruction Queue

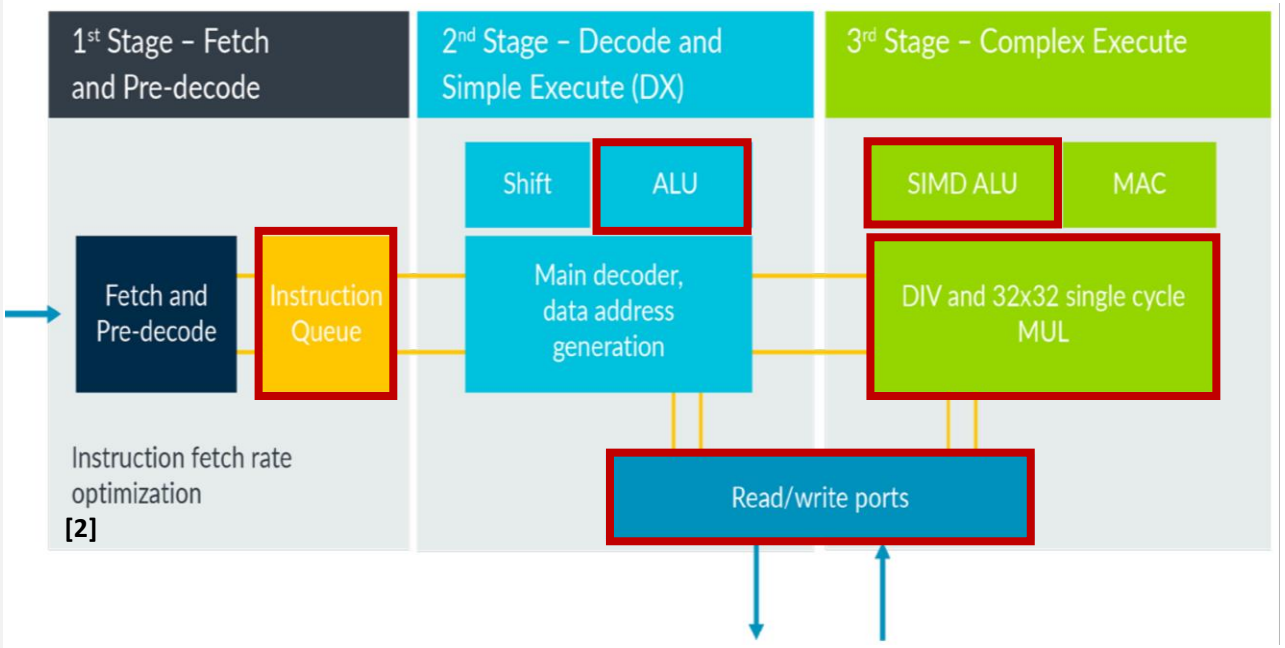
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2
- TLBs
- BP
- OoO



Arm Cortex-M33 Microarchitecture

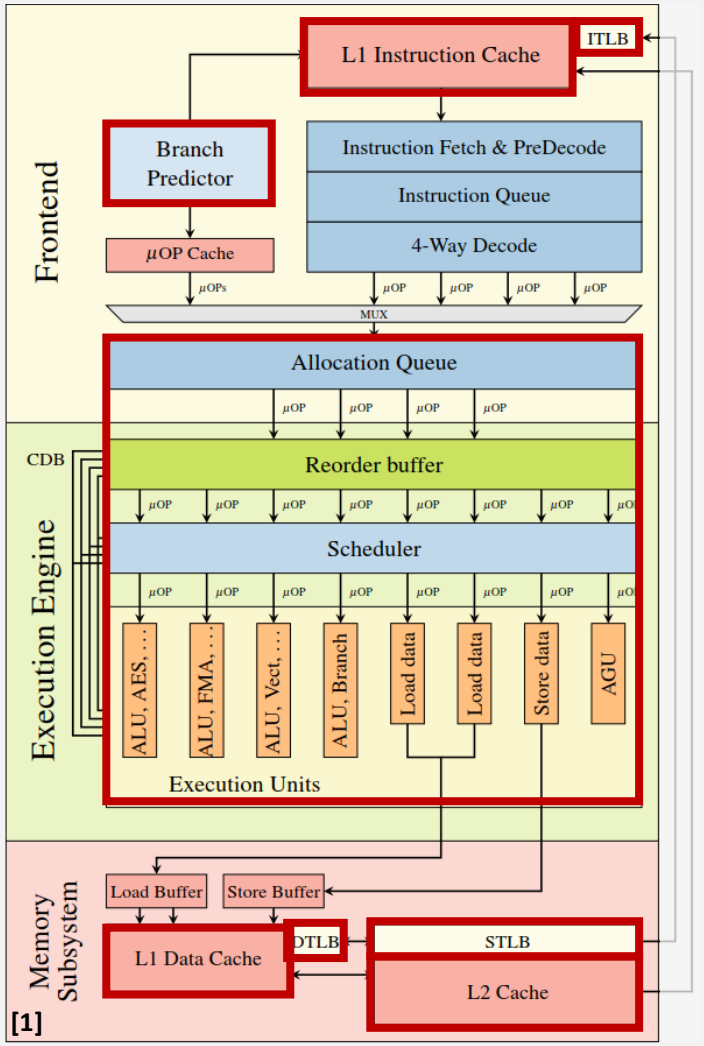


- Instruction Queue
- ALU, DIV and MUL

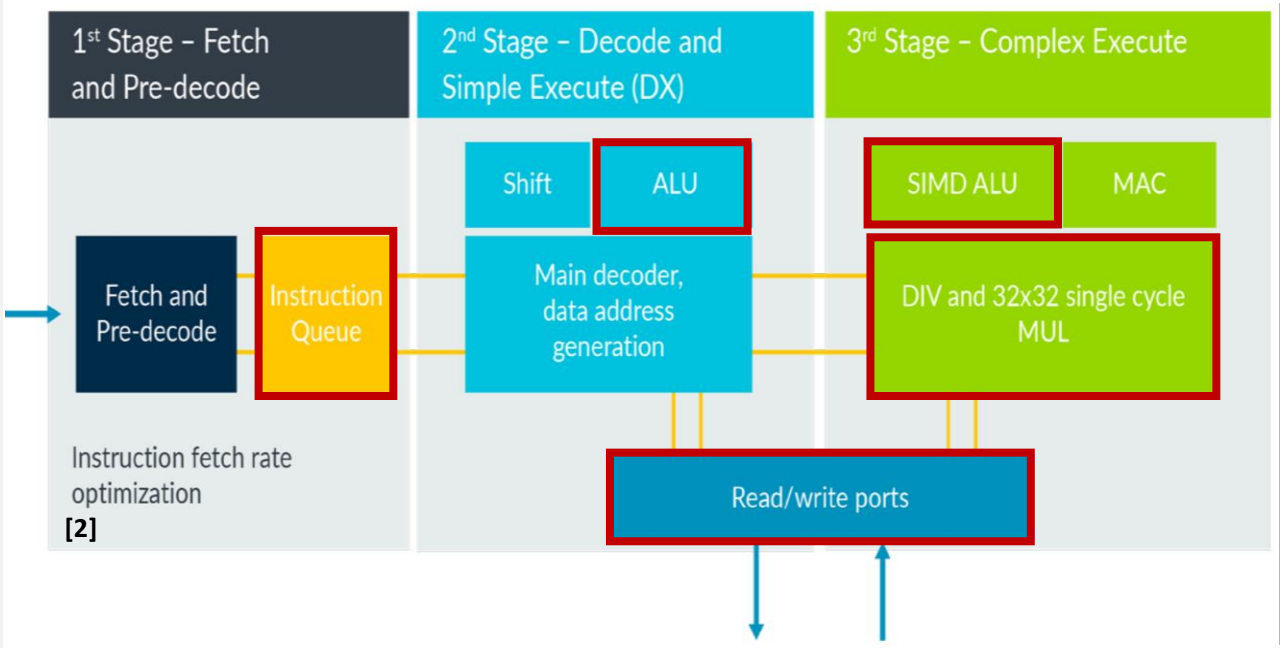
[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

Intel Skylake Microarchitecture

- L1-I
- L1-D
- L2
- TLBs
- BP
- OoO



Arm Cortex-M33 Microarchitecture



- Instruction Queue
- ALU, DIV and MUL
- Non-Cached Memory Accesses

[1] - Meltdown: Reading Kernel Memory from User Space
[2] - Arm Cortex-M33 Processor Datasheet

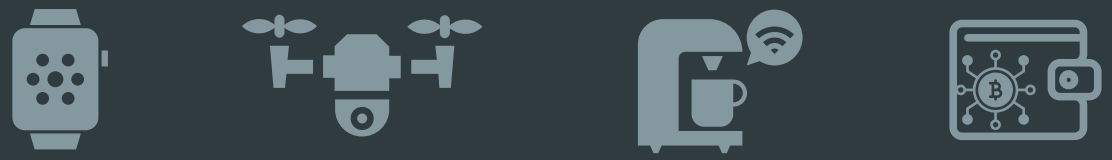
Which unique microarchitectural elements on MCUs may create new channels, and how can they be used to mount effective attacks?

Research Question

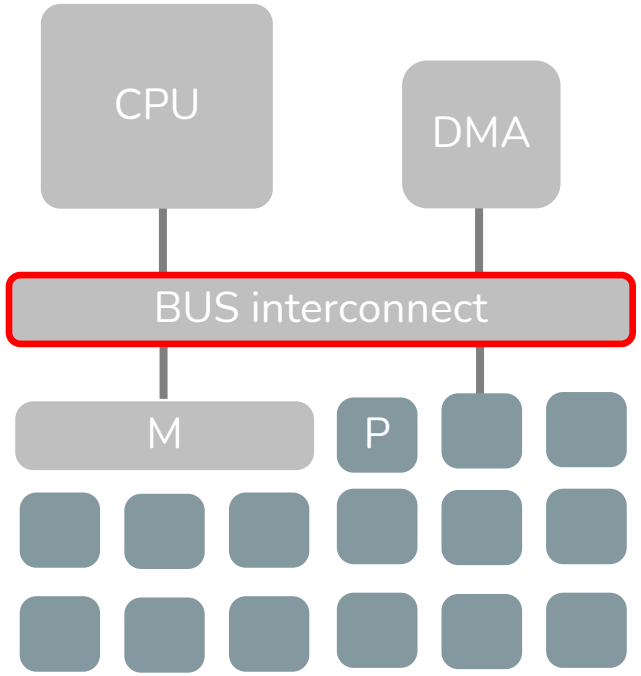
Hidden Threat

Novel Side-Channel Source: MCU Bus Interconnect

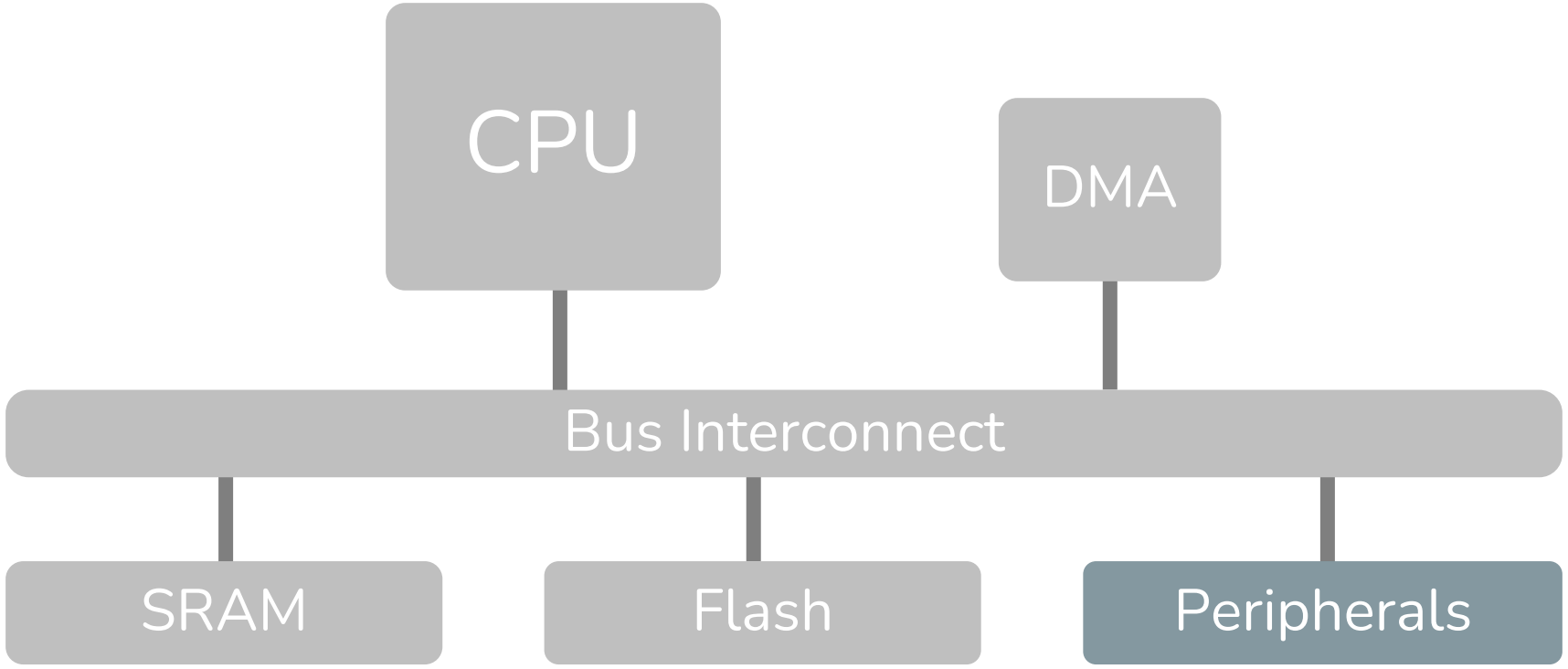
Microcontrollers



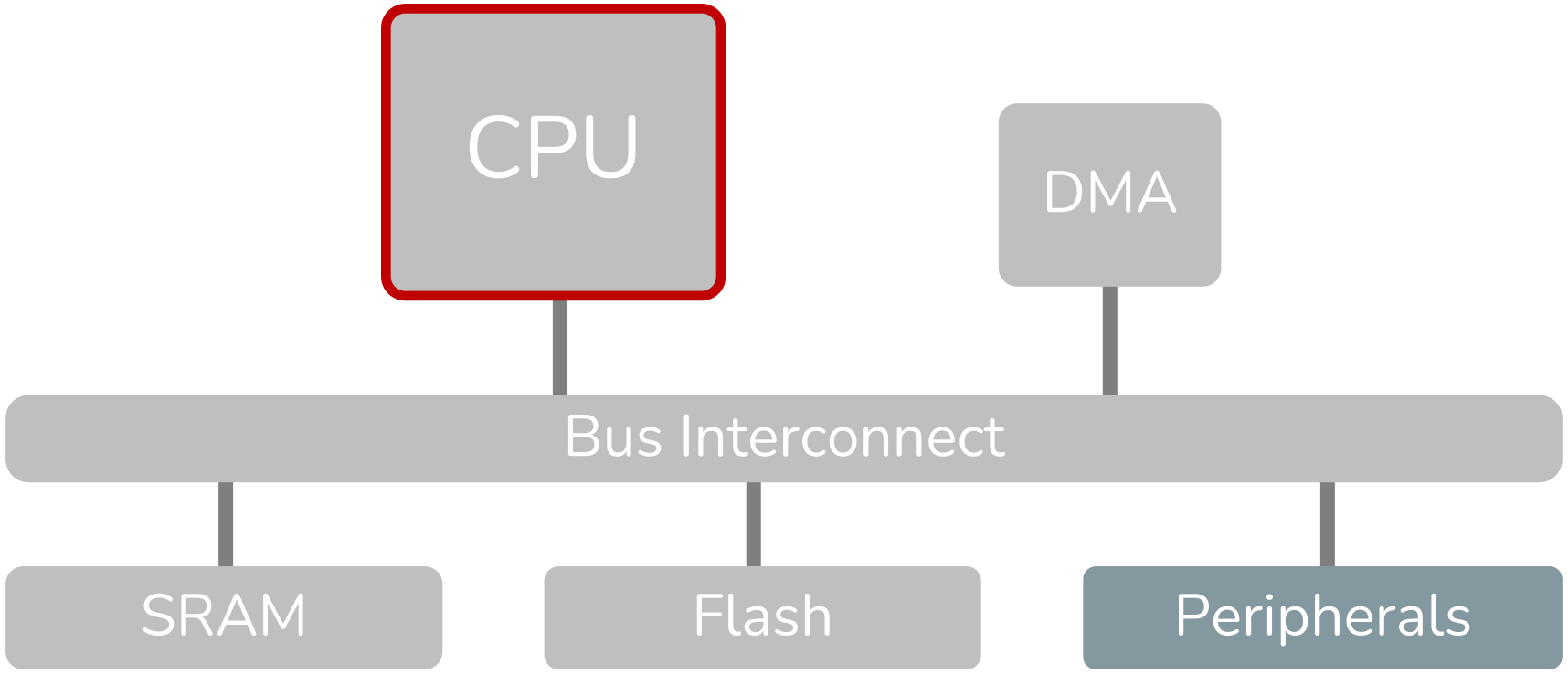
arm



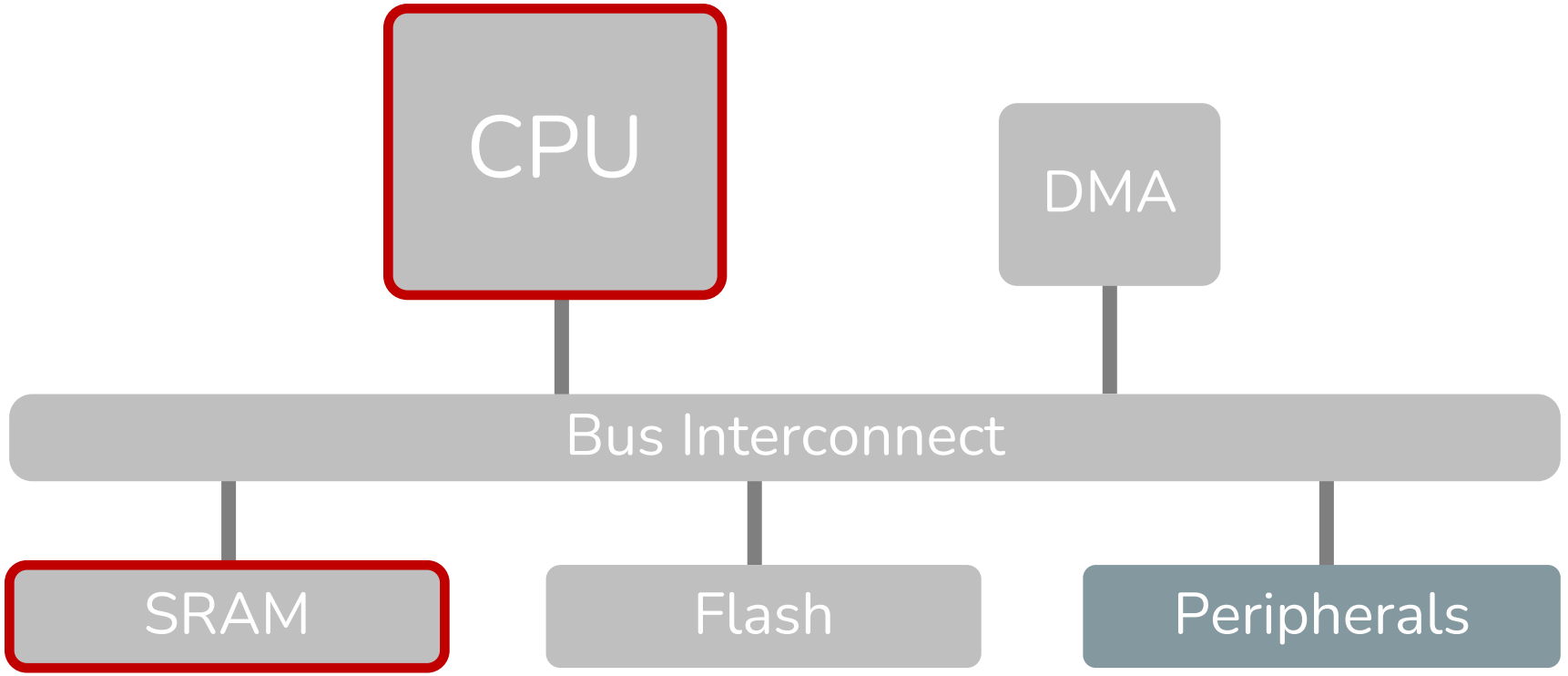
MCU Bus Interconnect as a Threat



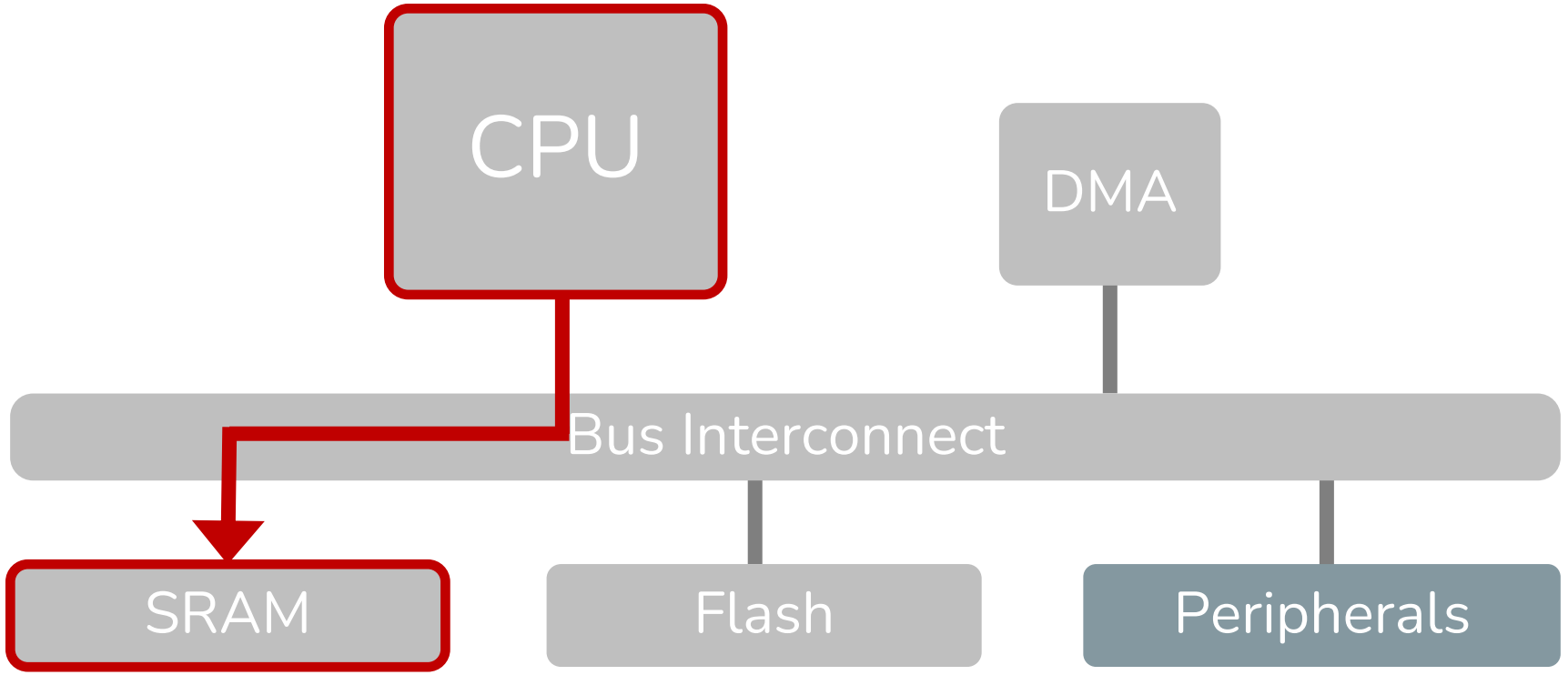
MCU Bus Interconnect as a Threat



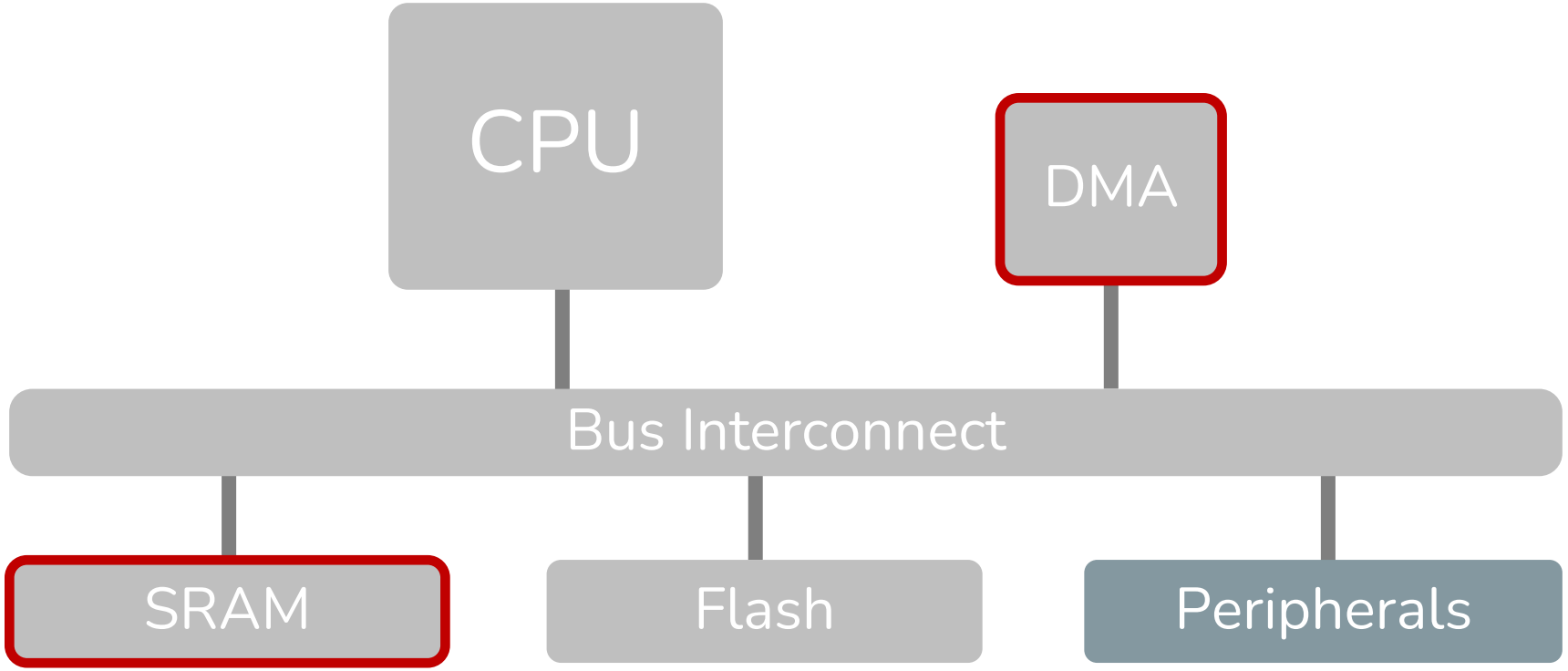
MCU Bus Interconnect as a Threat



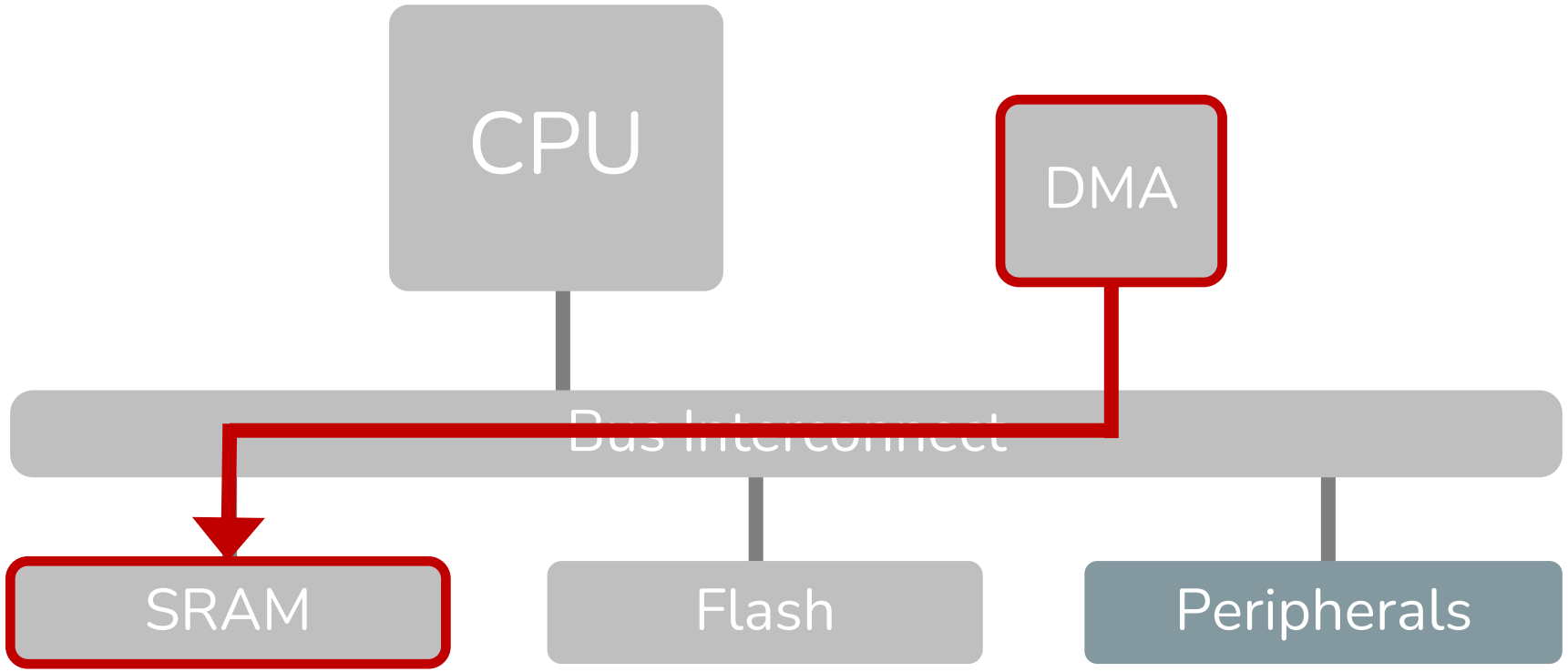
MCU Bus Interconnect as a Threat



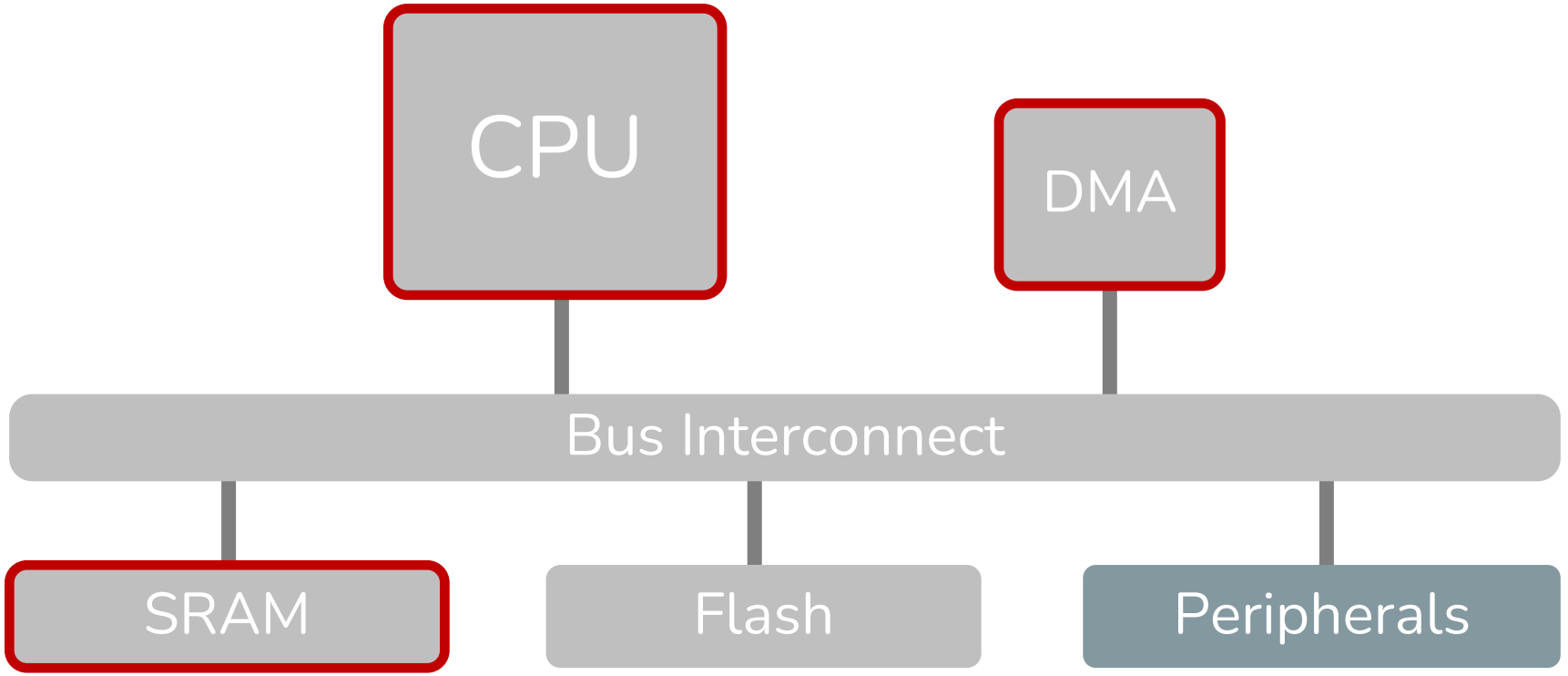
MCU Bus Interconnect as a Threat



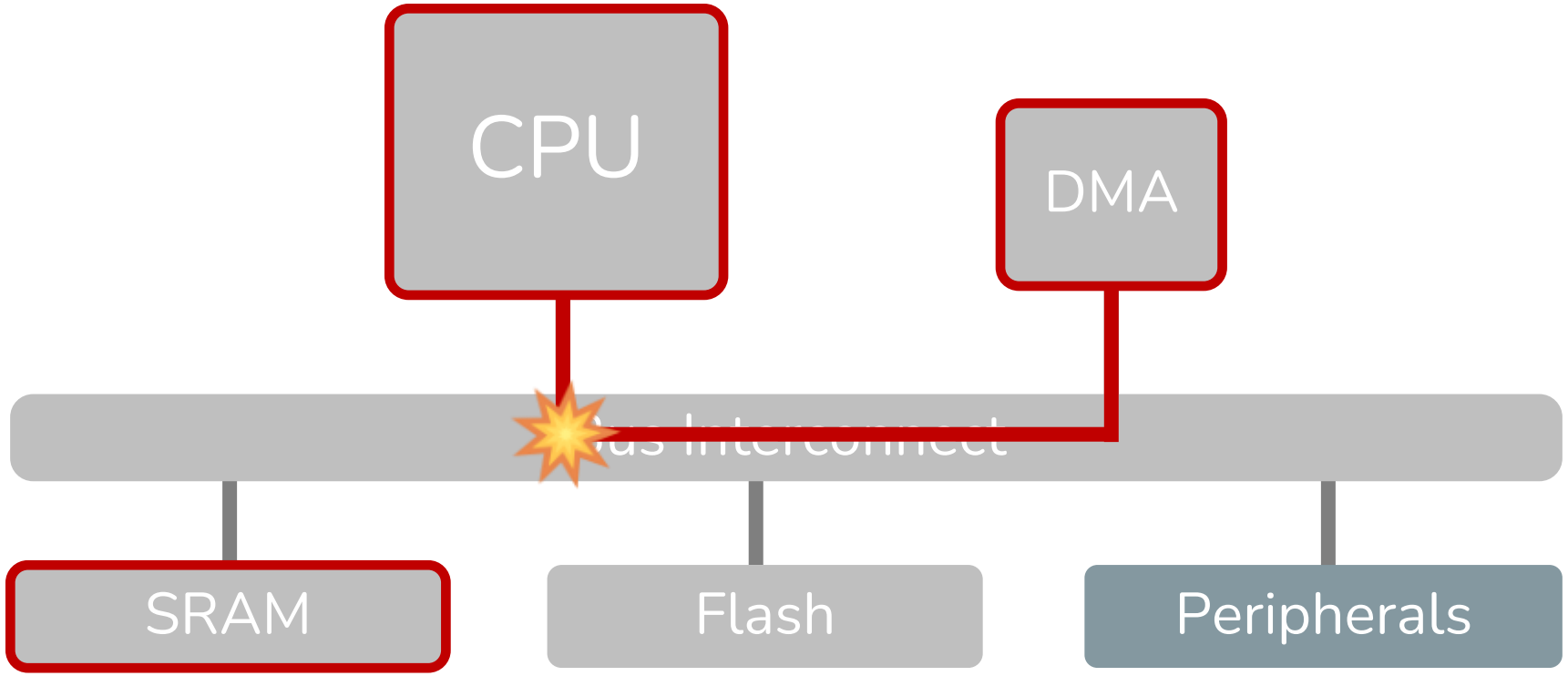
MCU Bus Interconnect as a Threat



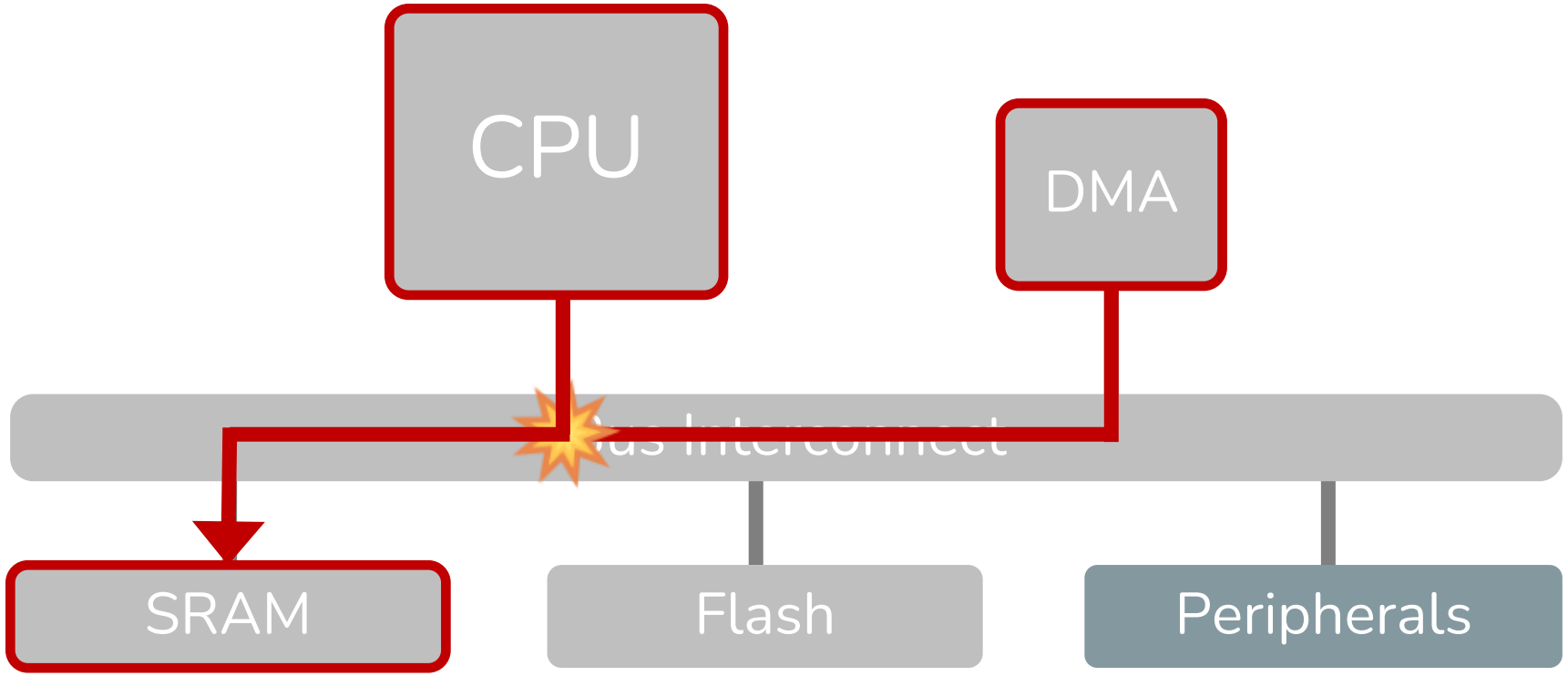
MCU Bus Interconnect as a Threat



MCU Bus Interconnect as a Threat



MCU Bus Interconnect as a Threat

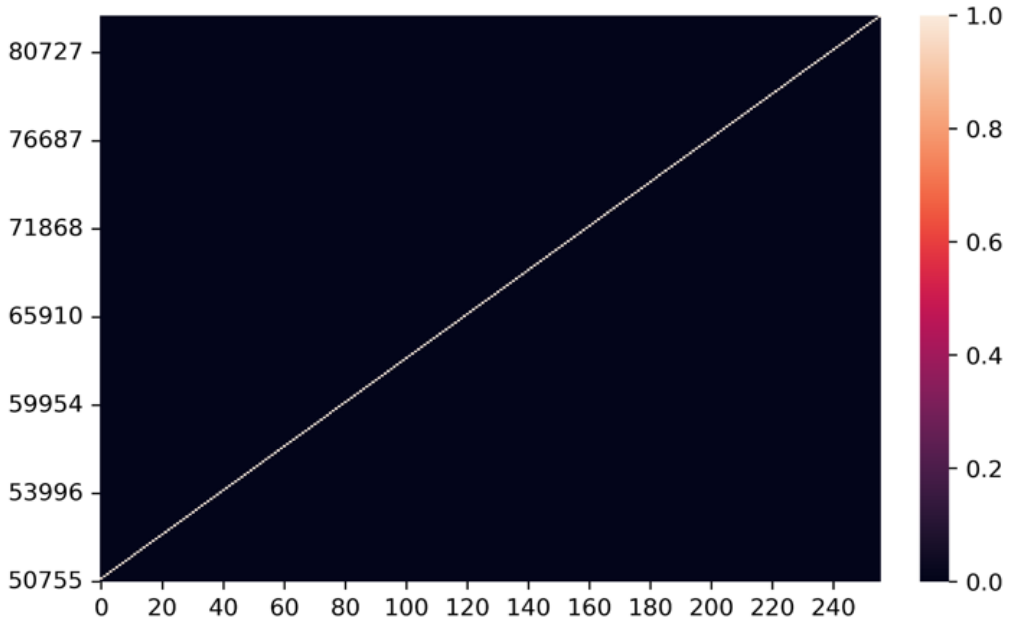


STM32L073 (M0+)

STM32L552 (M33)

STM32L412 (M4)

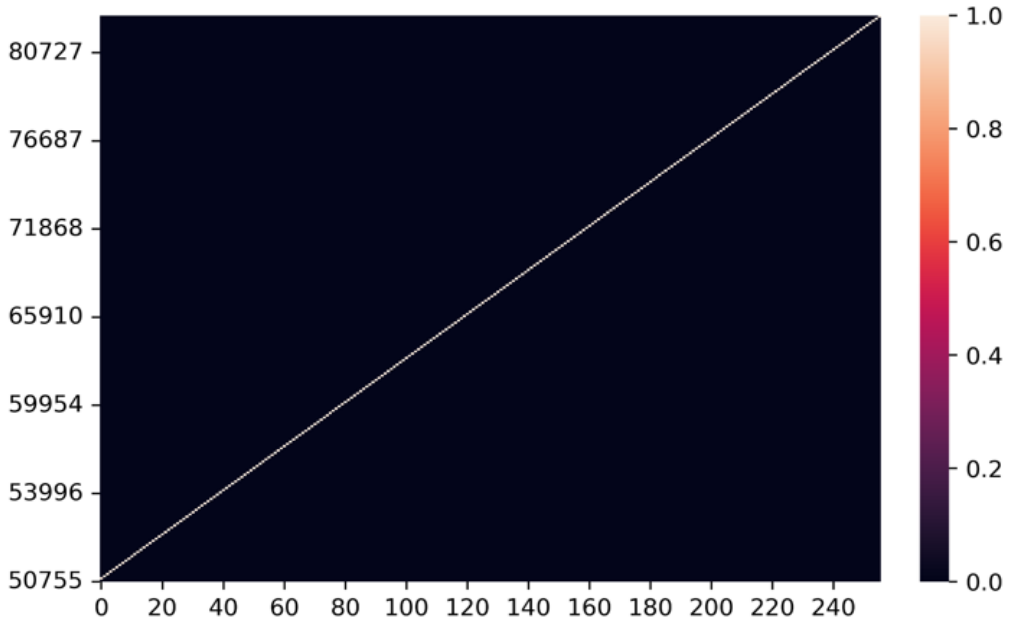
STM32L767 (M7)

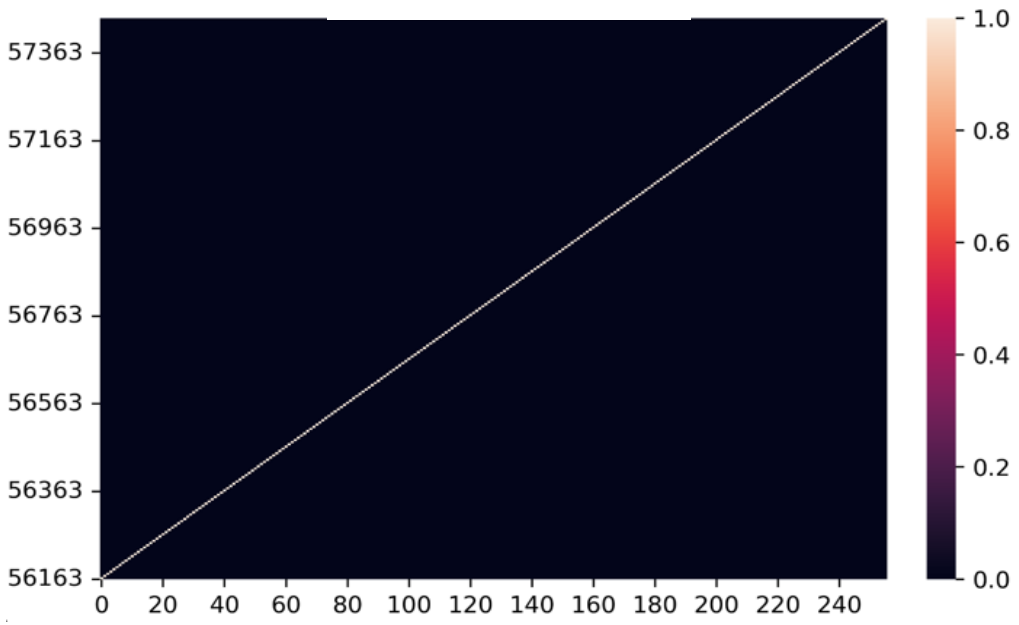


STM32L552 (M33)

STM32L412 (M4)

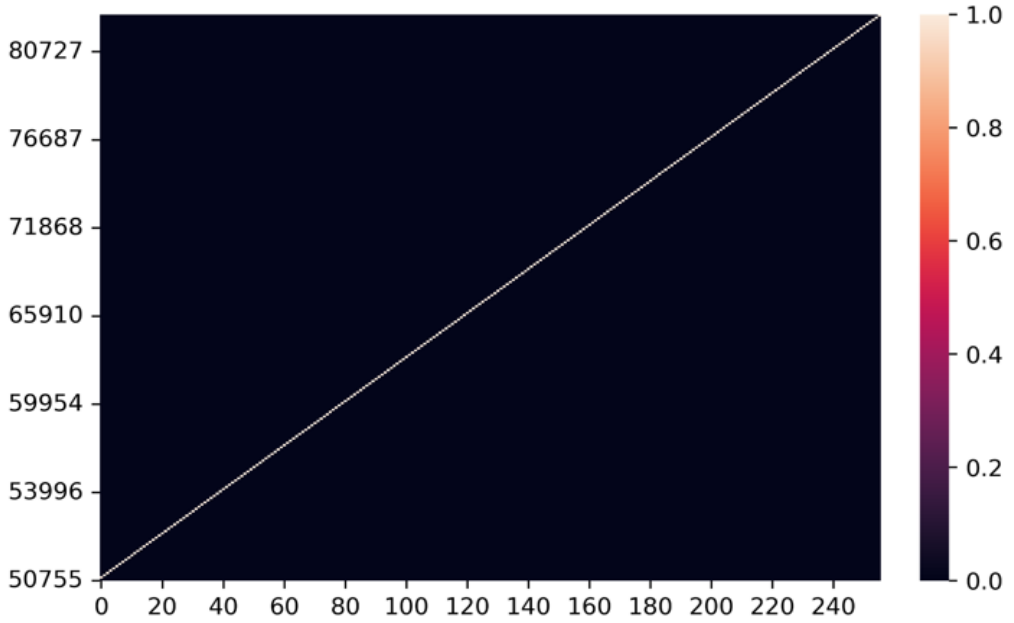
STM32L767 (M7)

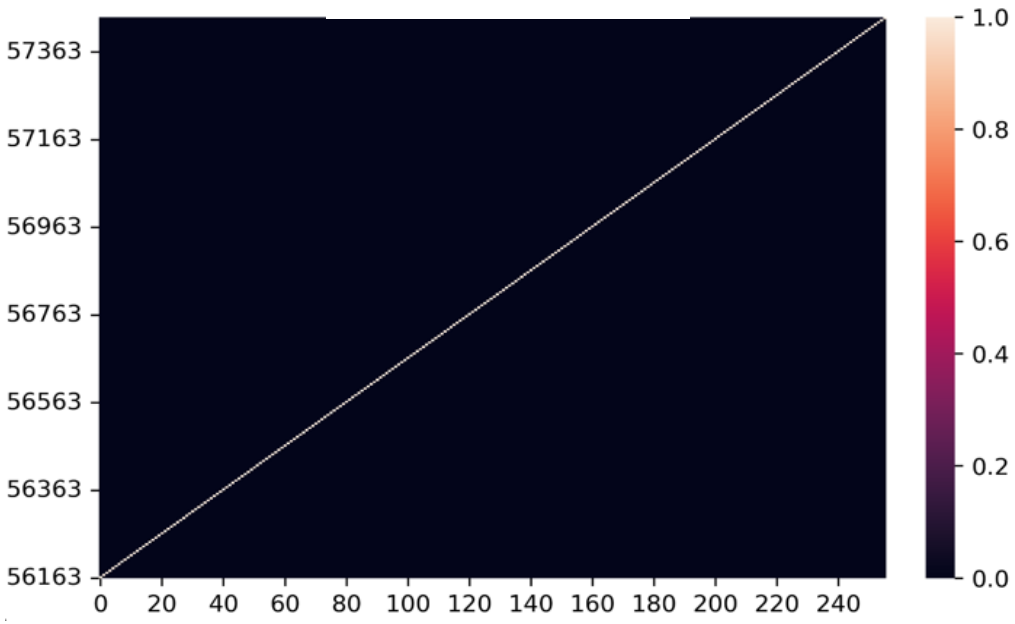


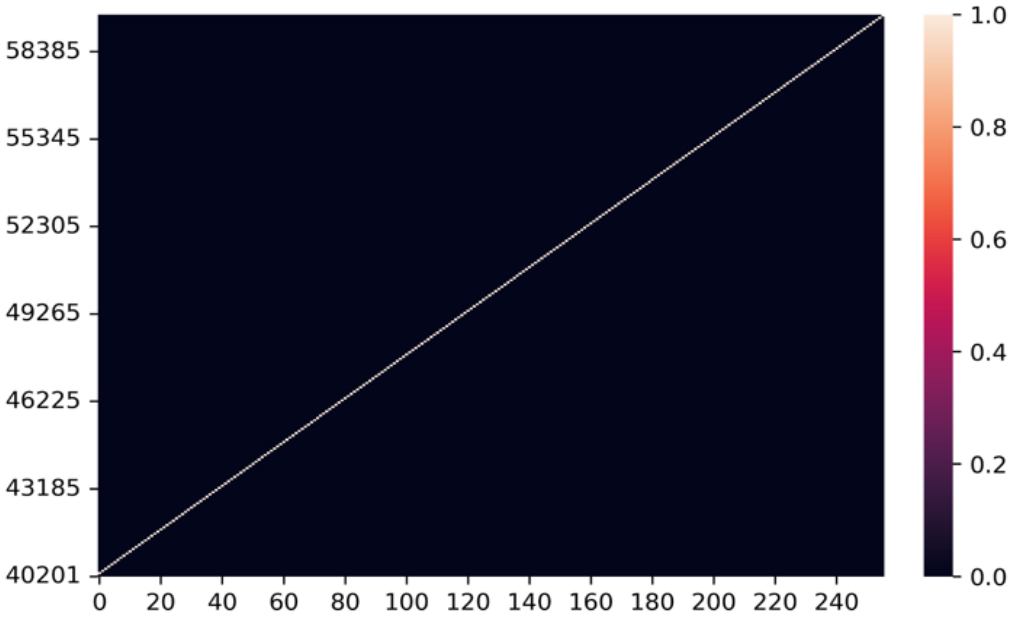


STM32L412 (M4)

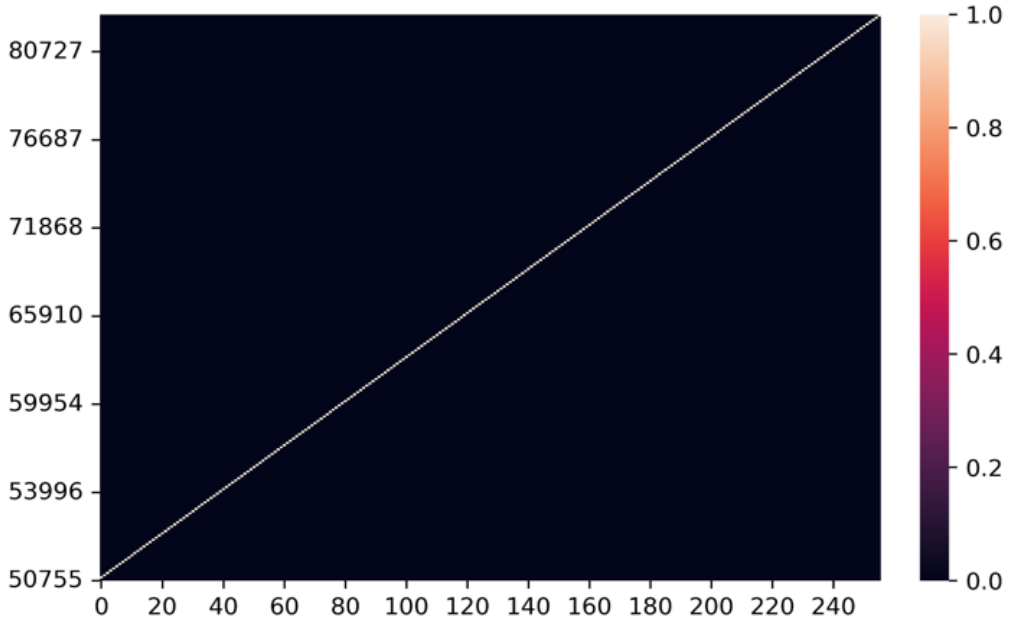
STM32L767 (M7)

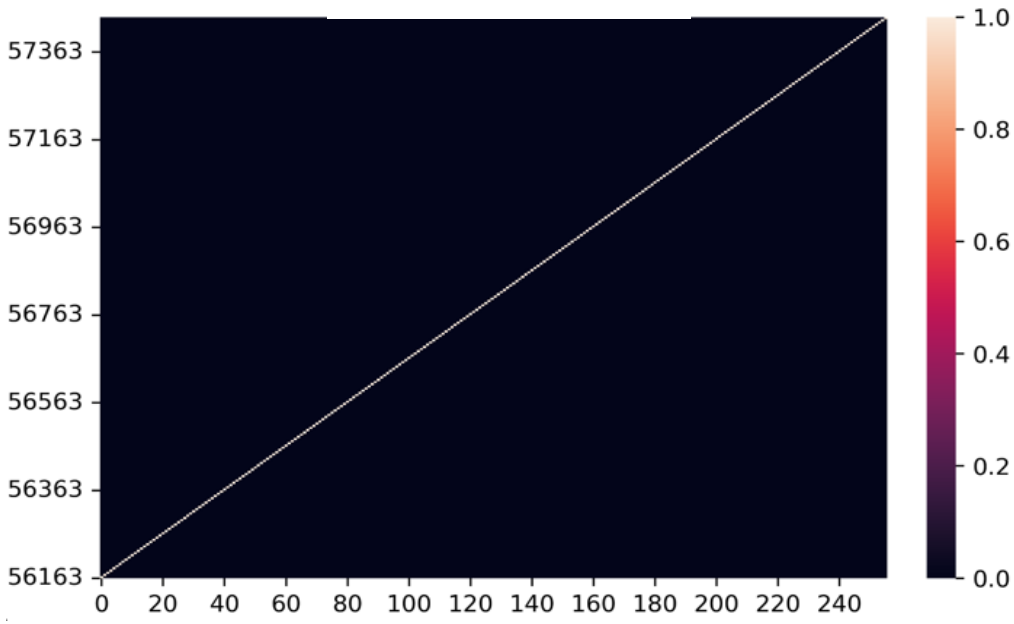


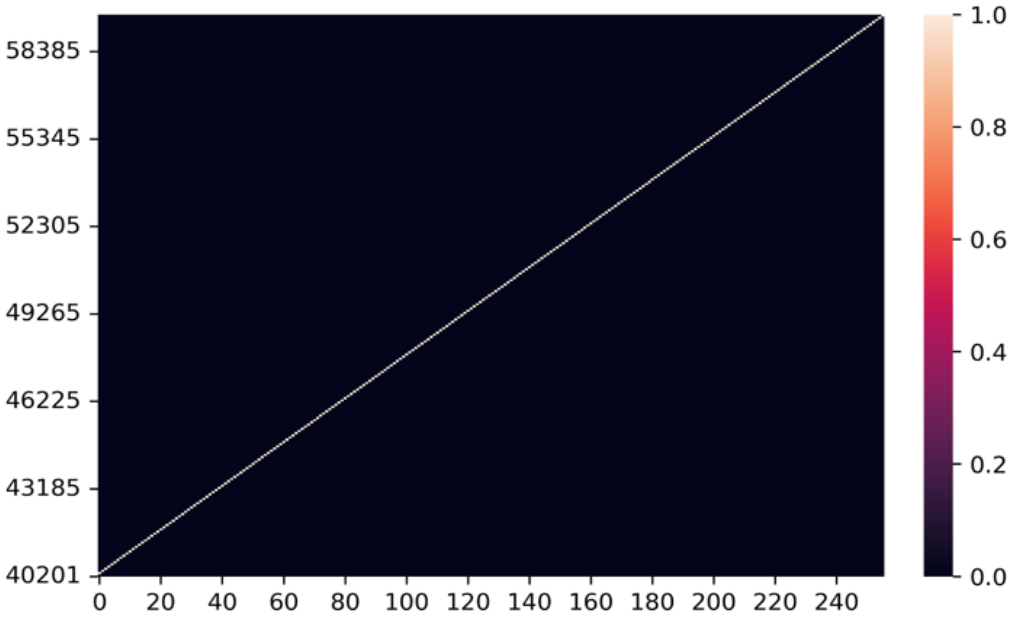


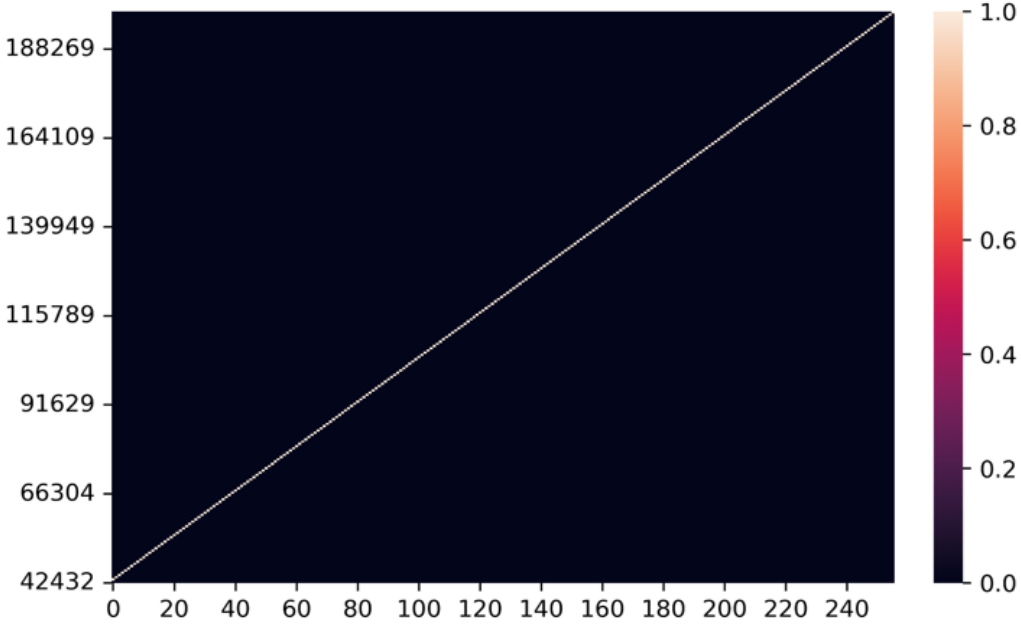


STM32L767 (M7)


80727
76687
71868
65910
59954
53996
50755
0 20 40 60 80 100 120 140 160 180 200 220 240
1.0
0.8
0.6
0.4
0.2
0.0


57363
57163
56963
56763
56563
56363
56163
0 20 40 60 80 100 120 140 160 180 200 220 240
1.0
0.8
0.6
0.4
0.2
0.0

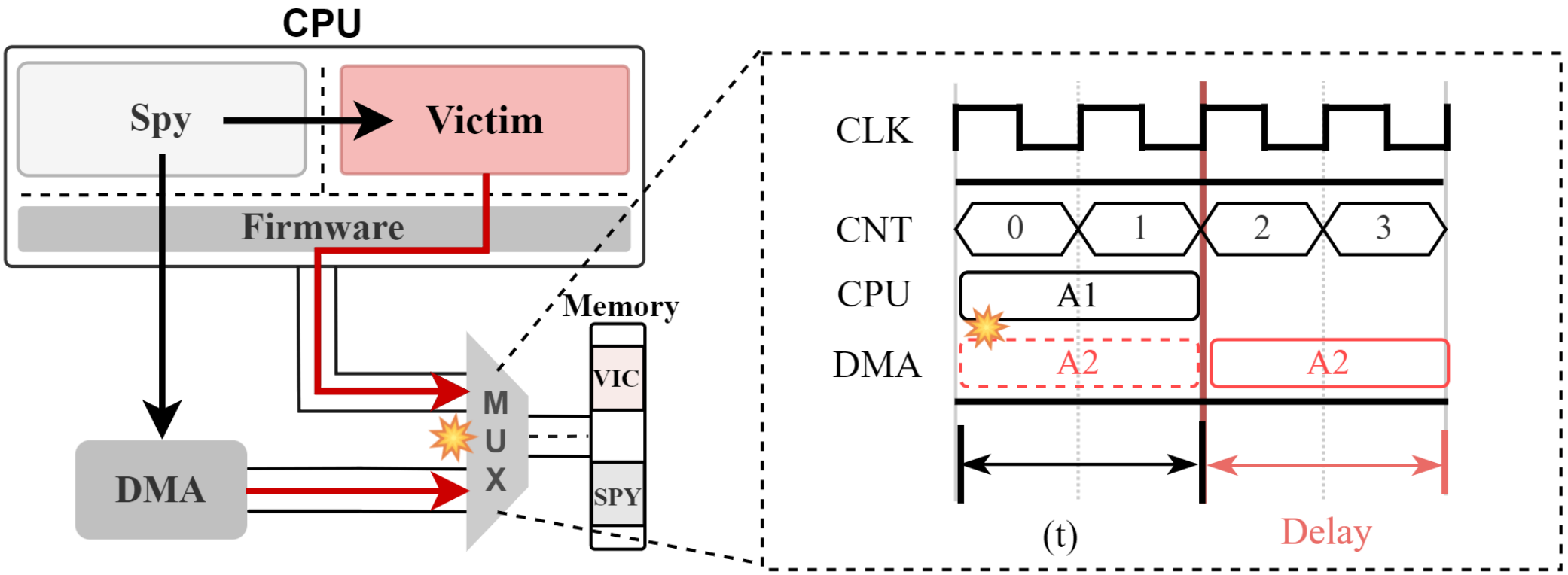

58385
55345
52305
49265
46225
43185
40201
0 20 40 60 80 100 120 140 160 180 200 220 240
1.0
0.8
0.6
0.4
0.2
0.0


188269
164109
139949
115789
91629
66304
42432
0 20 40 60 80 100 120 140 160 180 200 220 240
1.0
0.8
0.6
0.4
0.2
0.0

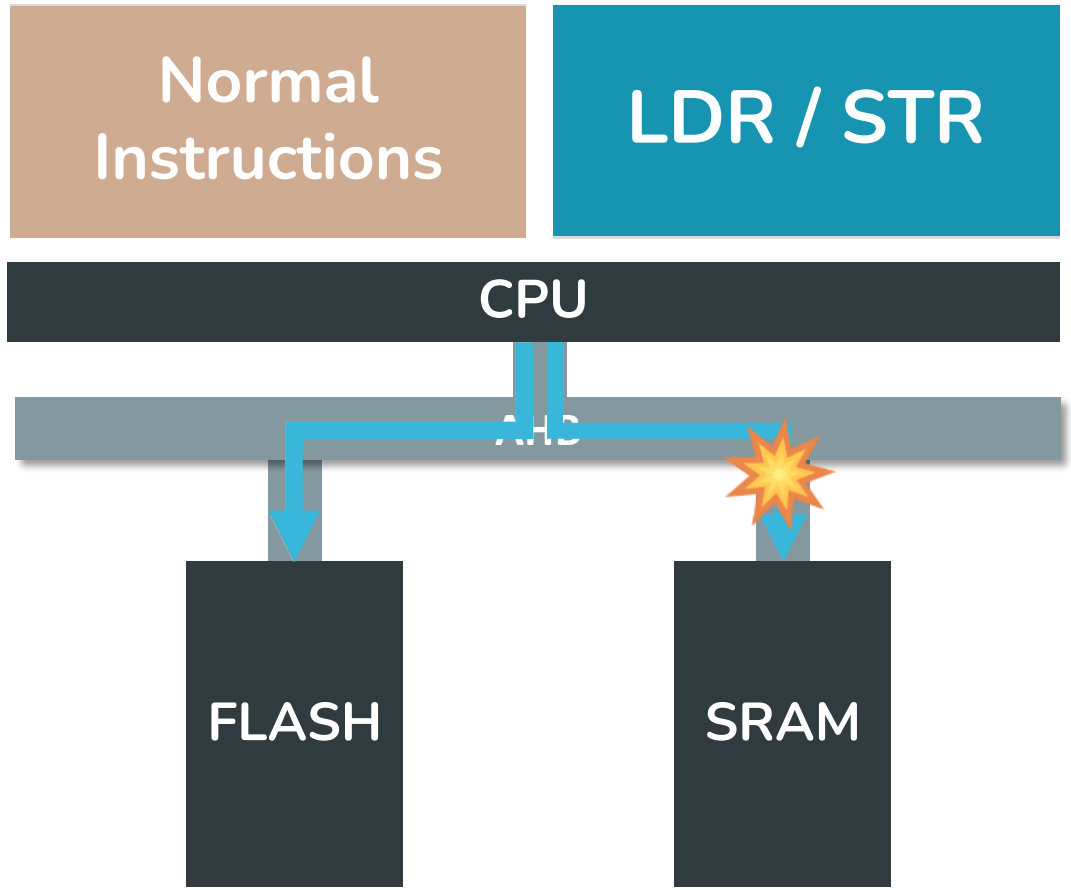
“Toy” Attack

Basic Attack Example

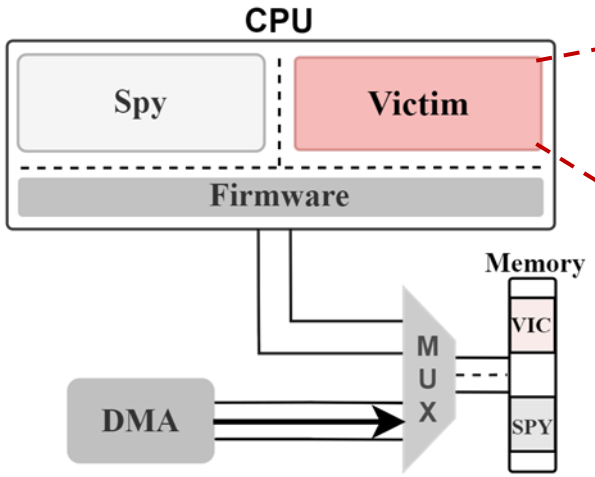
Attack Overview – The Basics



Attack Overview – The Basics



Attack Overview – Toy Example

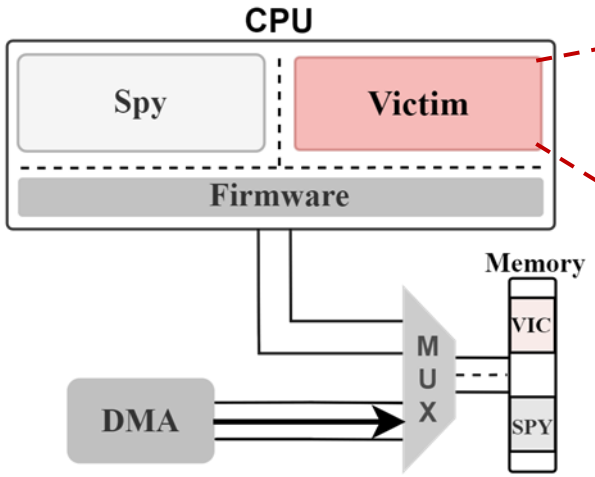



```
if(s==1)
  var=1;
else
  var=0;
```

```
      cmp   r3, #0
      beq.n ELSE
IF:   movs  r3, #1
      str   r3, [r7, #0]
      b.n   END
ELSE: movs  r3, #0
      str   r3, [r7, #0]
END:  nop
```

If-else statement compiled for Arm Cortex-M33 (-O0)

Attack Overview – Toy Example

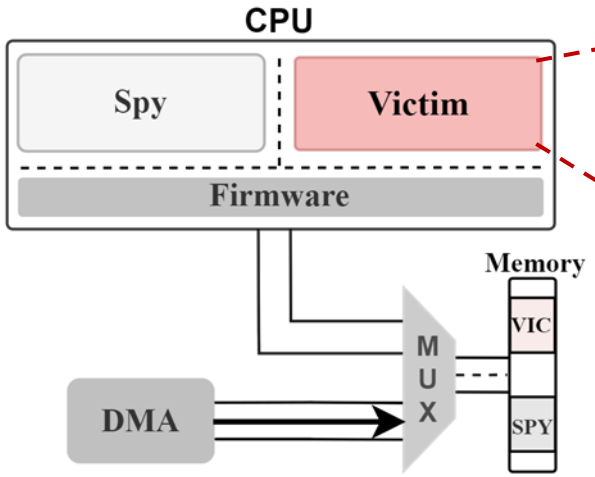



```
if(s==1)
  var=1;
else
  var=0;
```

```
      cmp   r3, #0
      beq.n ELSE
IF:   movs  r3, #1                1 clk
      str   r3, [r7, #0]          1 clk
      b.n   END                   2 clk
ELSE: movs  r3, #0
      str   r3, [r7, #0]
END:  nop                         1 clk
```

If-else statement compiled for Arm Cortex-M33 (-O0)

Attack Overview – Toy Example

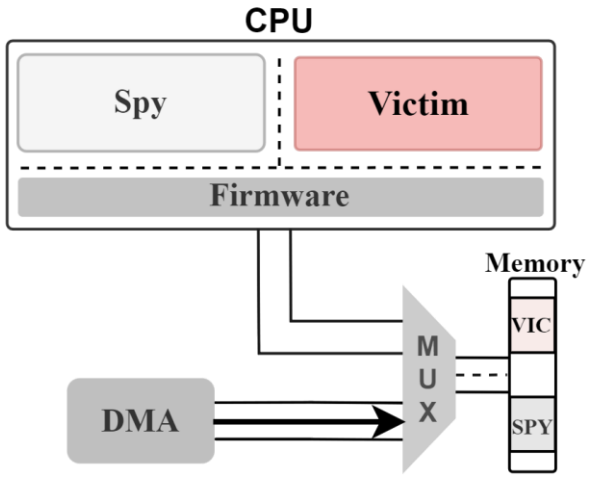



```
if(s==1)
  var=1;
else
  var=0;
```

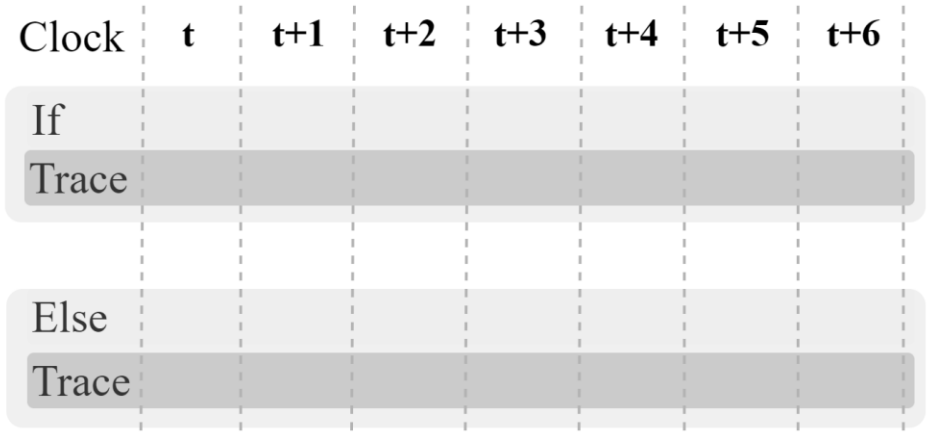
```
      cmp   r3, #0                    1 clk
      beq.n ELSE                      3 clk (else), 1 clk (if)
IF:   movs  r3, #1                    1 clk
      str   r3, [r7, #0]              1 clk
      b.n   END                       2 clk
ELSE: movs  r3, #0                    1 clk
      str   r3, [r7, #0]              1 clk
END:  nop                             1 clk
```

If-else statement compiled for Arm Cortex-M33 (-O0)

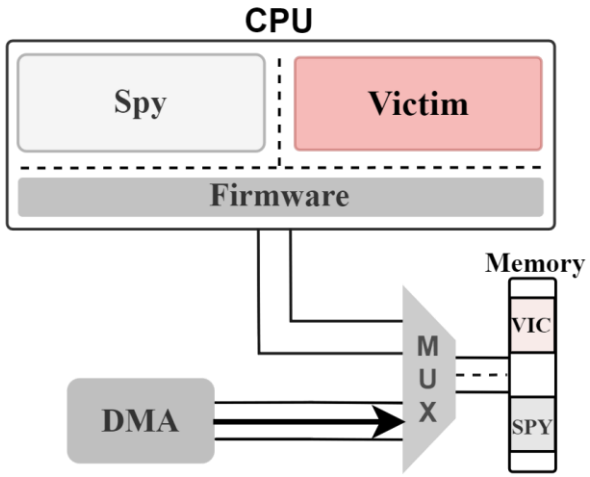
Attack Overview – Toy Example



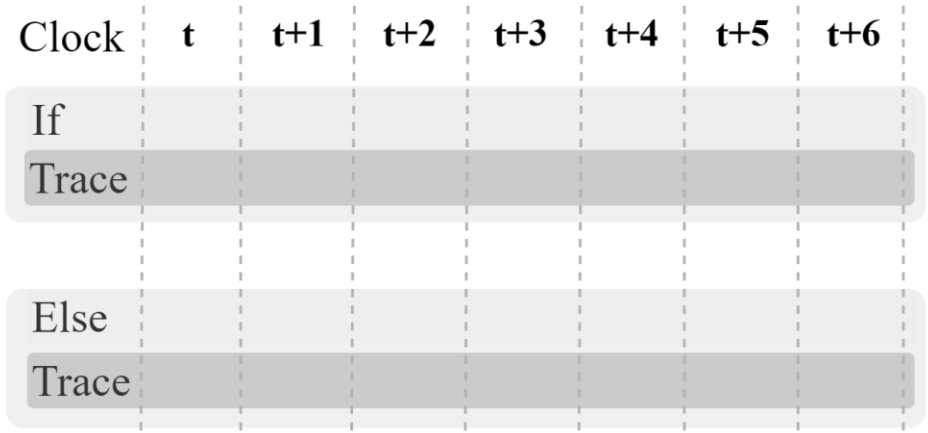

```
        cmp   r3, #0          1 clk
        beq.n ELSE            3 clk (else), 1 clk (if)
IF:     movs  r3, #1          1 clk
        str   r3, [r7, #0]    1 clk
        b.n   END             2 clk
ELSE:   movs  r3, #0          1 clk
        str   r3, [r7, #0]    1 clk
END:    nop                   1 clk
```



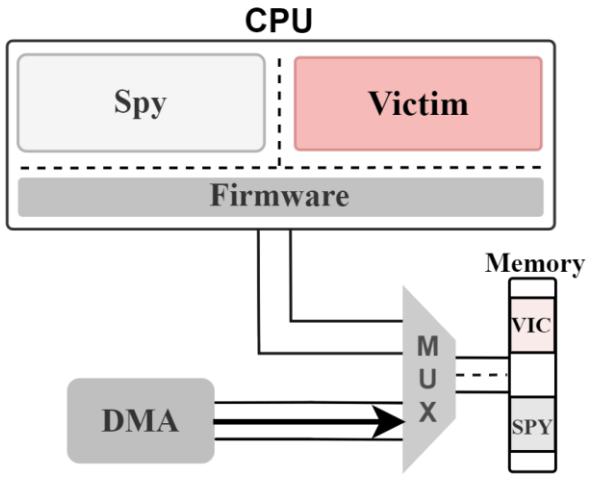
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE          3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```



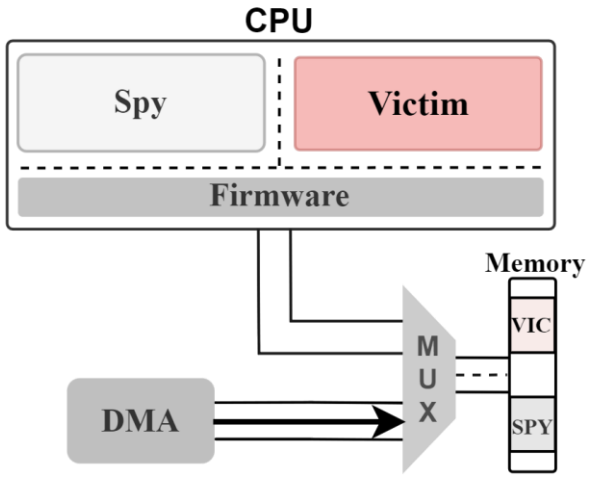
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp						
Trace	---						
Else							
Trace							

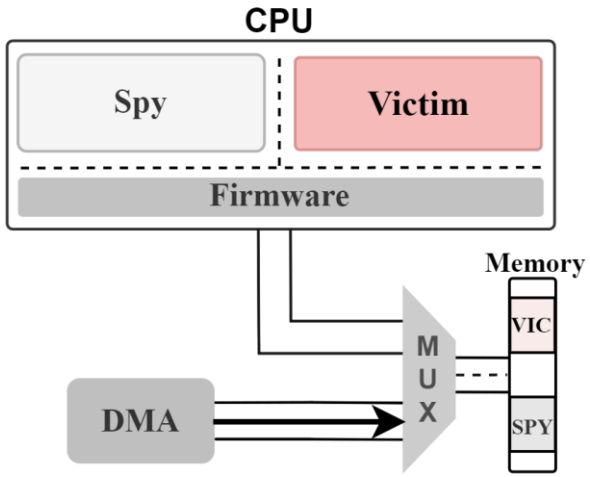
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq					
Trace	---	---					
Else							
Trace							

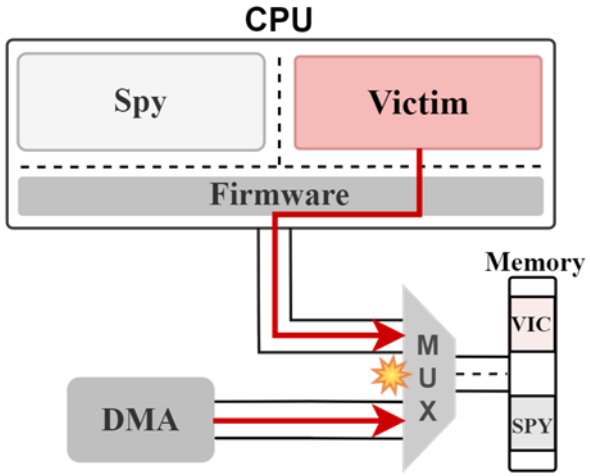
Attack Overview – Toy Example




```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n    END                       2 clk
ELSE:   movs   r3, #0                    1 clk
        str    r3, [r7, #0]              1 clk
END:    nop                              1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs				
Trace	---	---	---				
Else							
Trace							

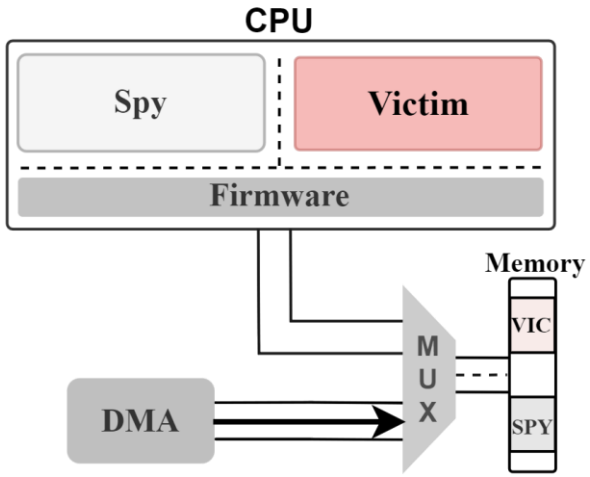
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str			
Trace	---	---	---	X			
Else							
Trace							

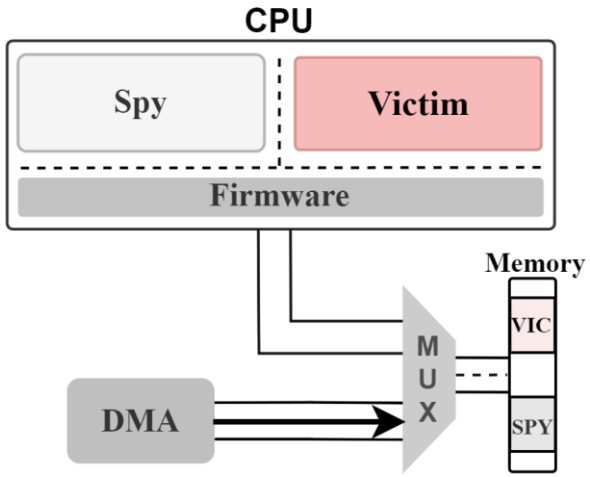
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b		
Trace	---	---	---	X	---		
Else							
Trace							

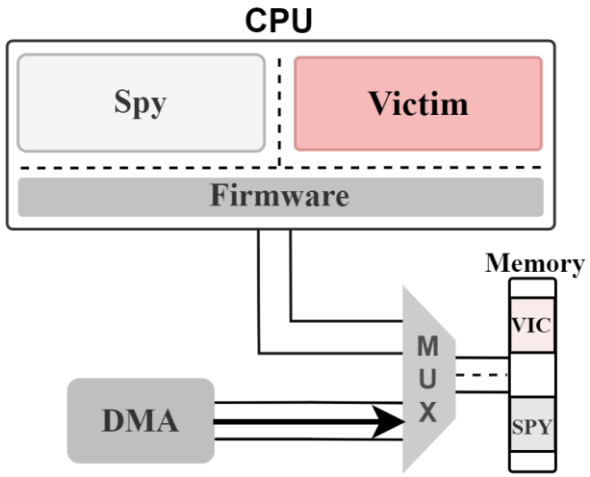
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	
Trace	---	---	---	X	---	---	
Else							
Trace							

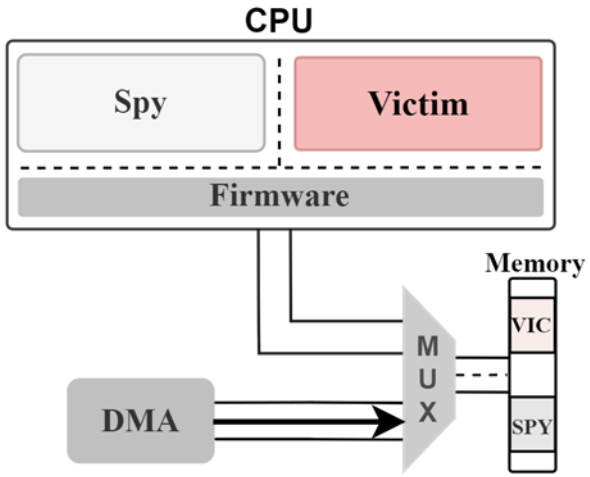
Attack Overview – Toy Example




```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n    END                       2 clk
ELSE:   movs   r3, #0                    1 clk
        str    r3, [r7, #0]              1 clk
END:    nop                              1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else							
Trace							

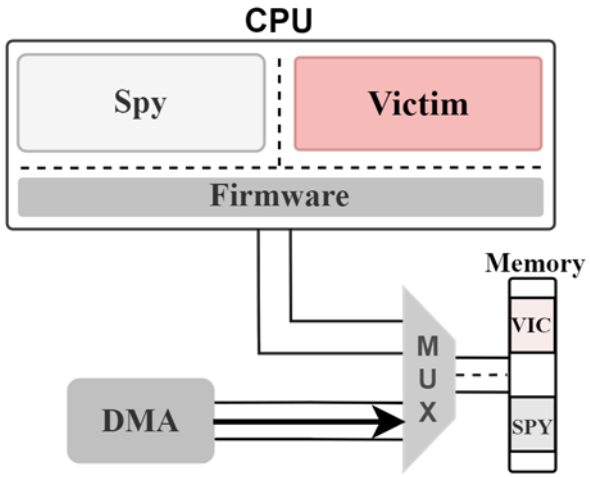
Attack Overview – Toy Example




```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n    END                       2 clk
ELSE:   movs   r3, #0                    1 clk
        str    r3, [r7, #0]              1 clk
END:    nop                              1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else							
Trace							

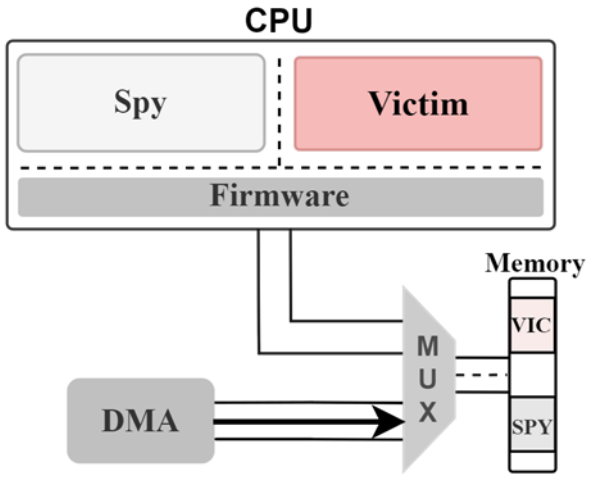
Attack Overview – Toy Example




```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n    END                       2 clk
ELSE:   movs   r3, #0                    1 clk
        str    r3, [r7, #0]              1 clk
END:    nop                              1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp						
Trace	---						

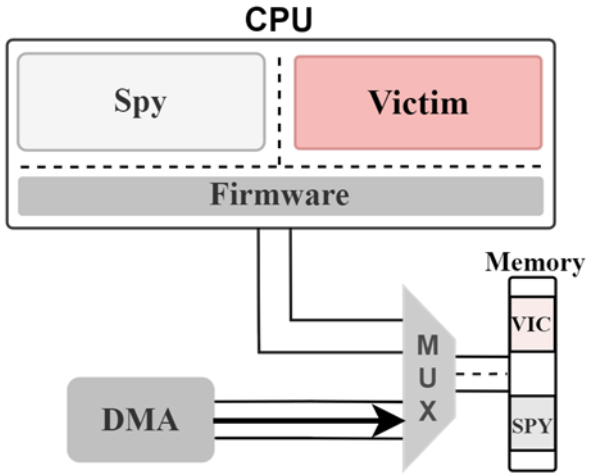
Attack Overview – Toy Example




```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n    END                       2 clk
ELSE:   movs   r3, #0                    1 clk
        str    r3, [r7, #0]              1 clk
END:    nop                              1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp						
Trace	---						

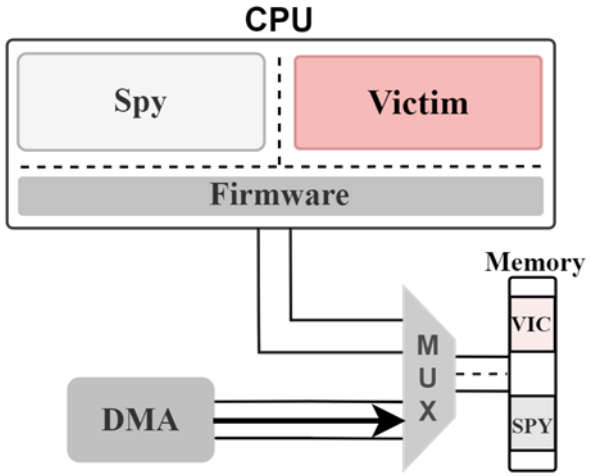
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq					
Trace	---	---					

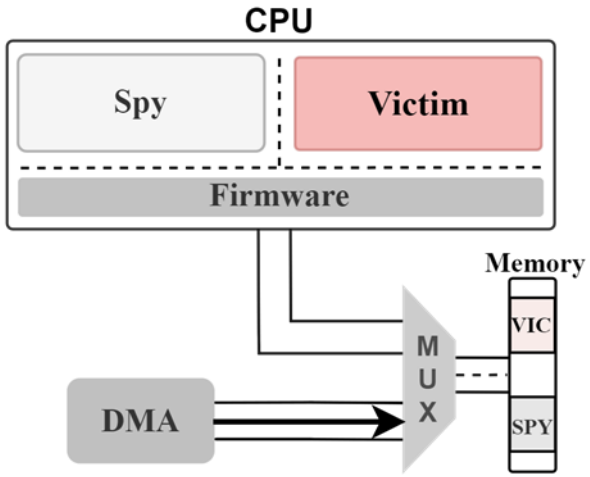
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq				
Trace	---	---	---				

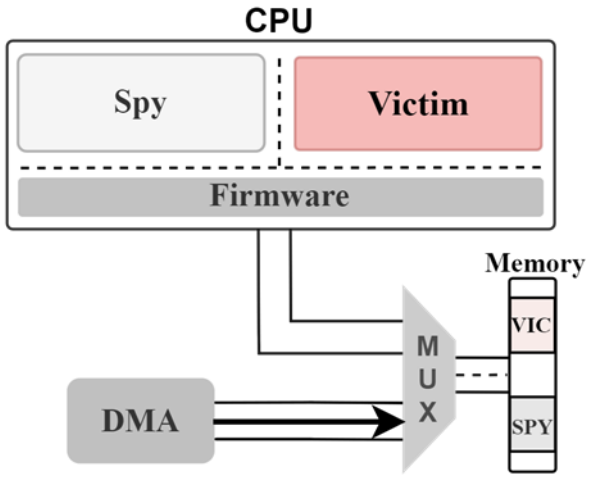
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq			
Trace	---	---	---	---			

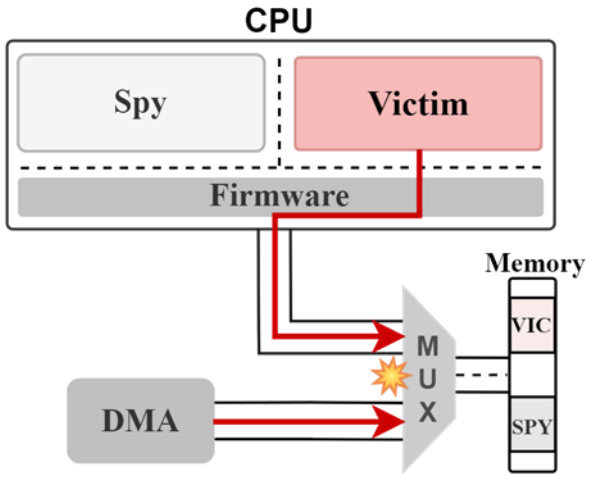
Attack Overview – Toy Example




```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n    END                       2 clk
ELSE:   movs   r3, #0                    1 clk
        str    r3, [r7, #0]              1 clk
END:    nop                              1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs		
Trace	---	---	---	---	---		

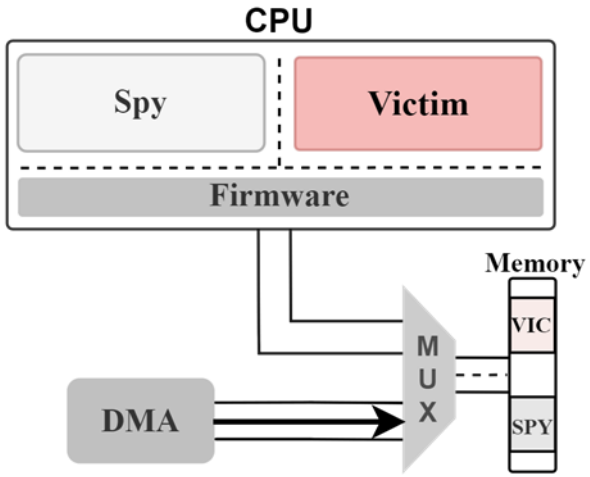
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs	str	
Trace	---	---	---	---	---	X	

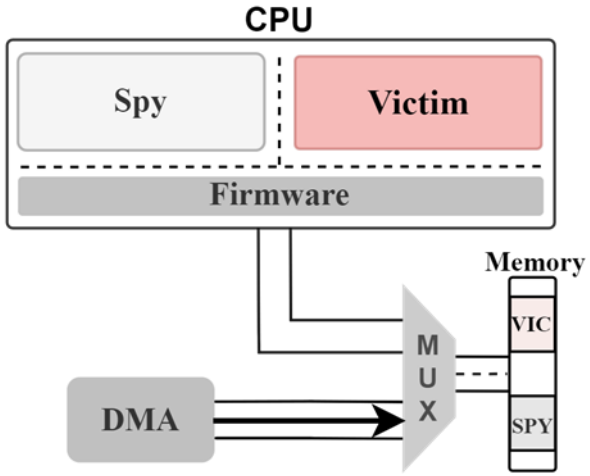
Attack Overview – Toy Example



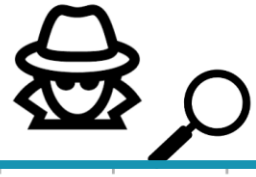

```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n    END                       2 clk
ELSE:   movs   r3, #0                    1 clk
        str    r3, [r7, #0]              1 clk
END:    nop                              1 clk
```

Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs	str	nop
Trace	---	---	---	---	---	X	---

Attack Overview – Toy Example

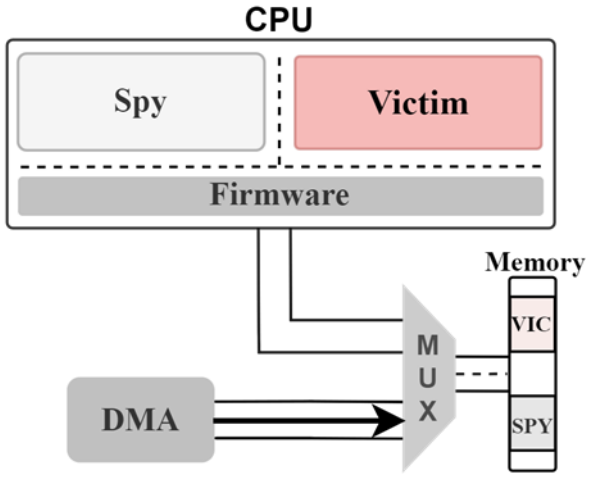



```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

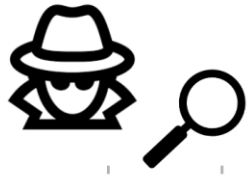


Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs	str	nop
Trace	---	---	---	---	---	X	---

Attack Overview – Toy Example

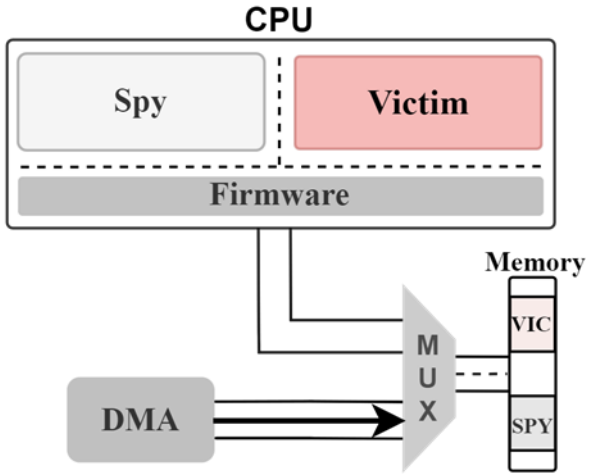



```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

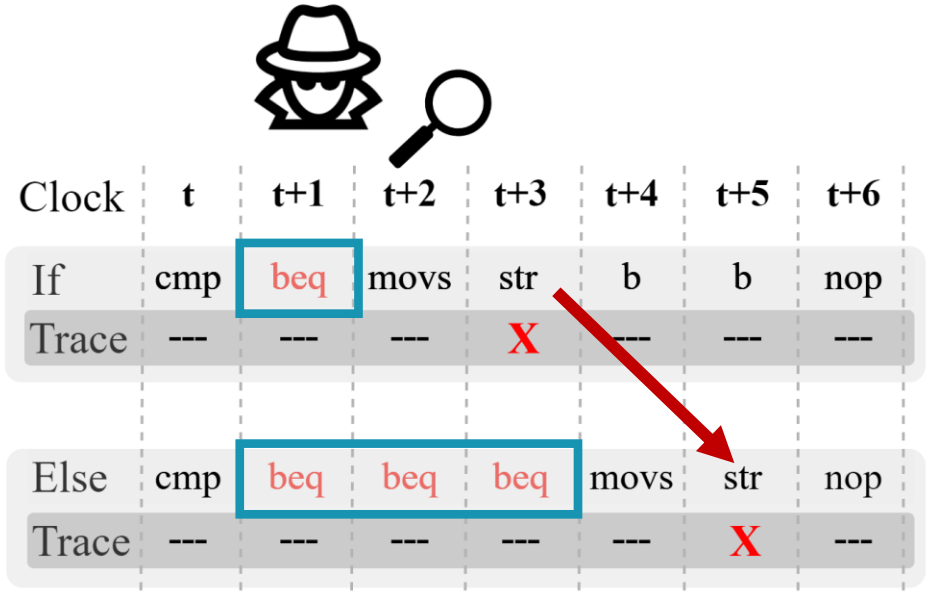


Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs	str	nop
Trace	---	---	---	---	---	X	---

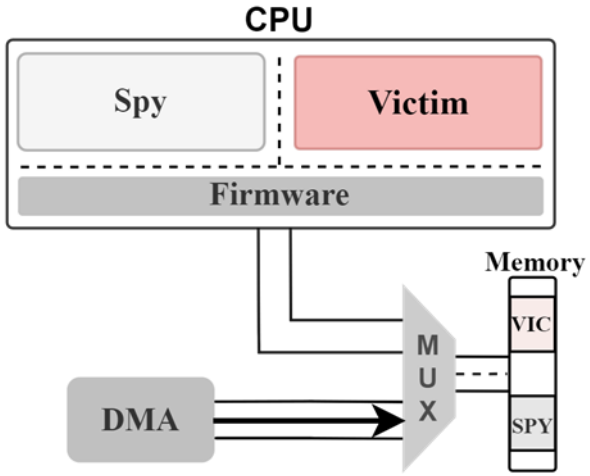
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```



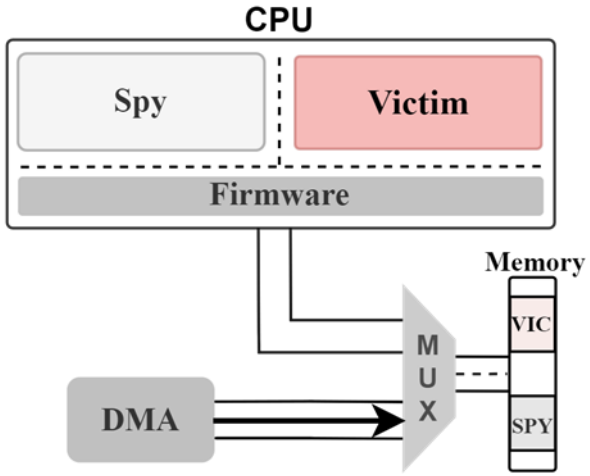
Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

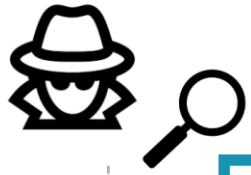
Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs	str	nop
Trace	---	---	---	---	---	X	---

Attack Overview – Toy Example



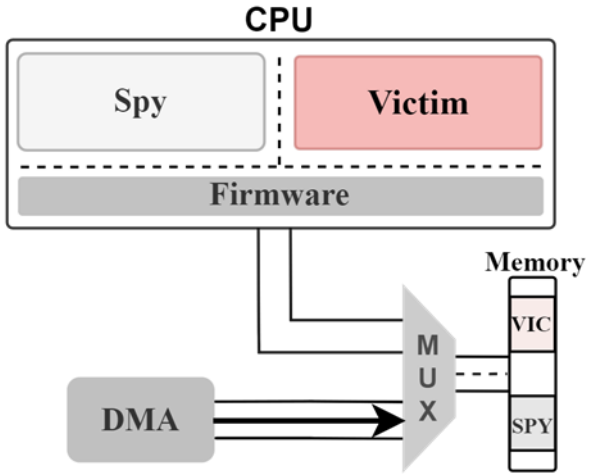

```
        cmp    r3, #0                    1 clk
        beq.n  ELSE                      3 clk (else), 1 clk (if)
IF:     movs   r3, #1                    1 clk
        str    r3, [r7, #0]              1 clk
        b.n                              2 clk
ELSE:   mov                              1 clk
        str                              1 clk
END:    nop                              1 clk
```

```
if(s==1)
  var=1;
else
  var=0;
```



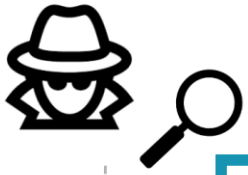
Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs	str	nop
Trace	---	---	---	---	---	X	---

Attack Overview – Toy Example




```
        cmp   r3, #0                    1 clk
        beq.n ELSE                      3 clk (else), 1 clk (if)
IF:     movs  r3, #1                    1 clk
        str   r3, [r7, #0]              1 clk
        b.n   END                       2 clk
ELSE:   movs  r3, #0                    1 clk
        str   r3, [r7, #0]              1 clk
END:    nop                             1 clk
```

```
if(s==1)
   var=1;
else
   var=0;
```

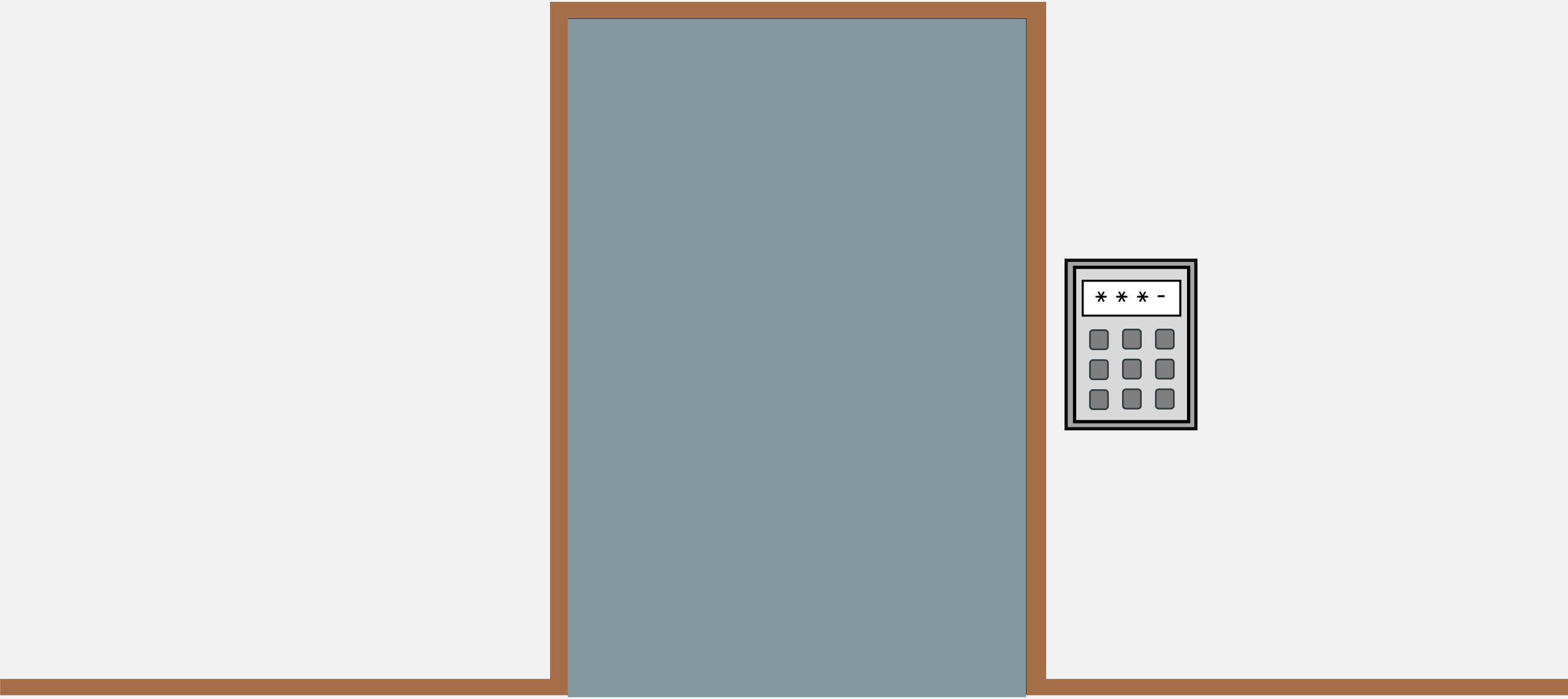


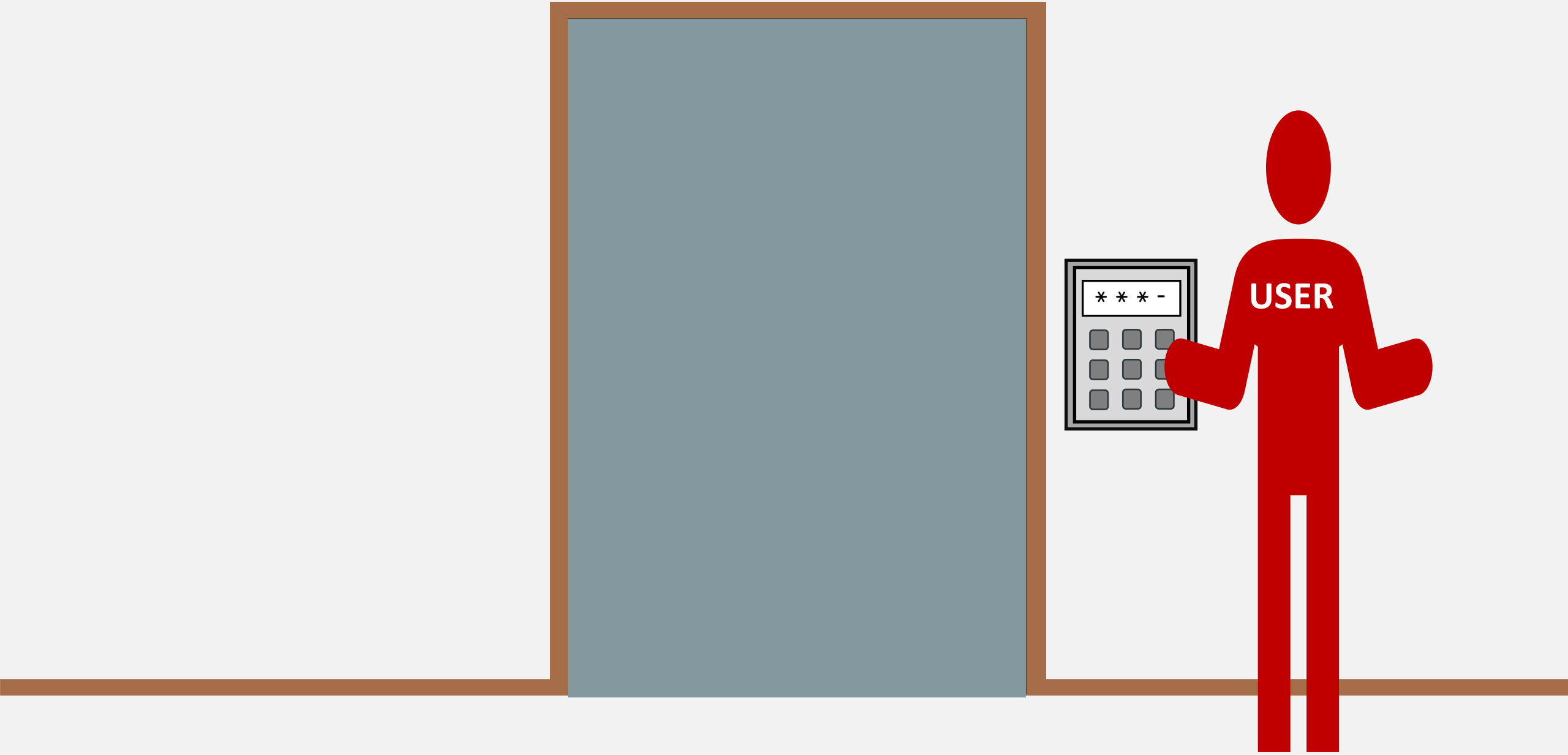
Clock	t	t+1	t+2	t+3	t+4	t+5	t+6
If	cmp	beq	movs	str	b	b	nop
Trace	---	---	---	X	---	---	---
Else	cmp	beq	beq	beq	movs	str	nop
Trace	---	---	---	---	---	X	---

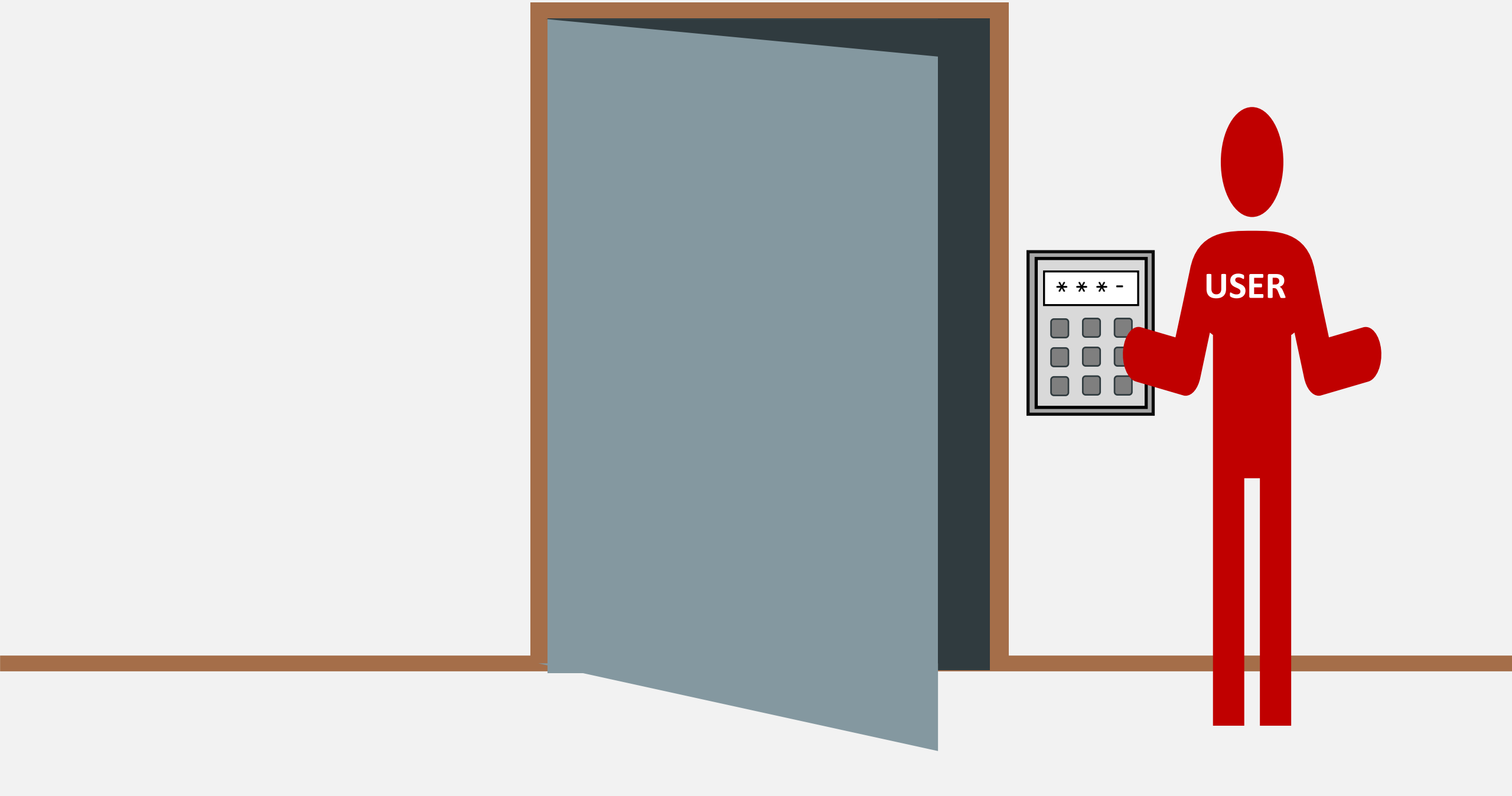
SECRET = 1

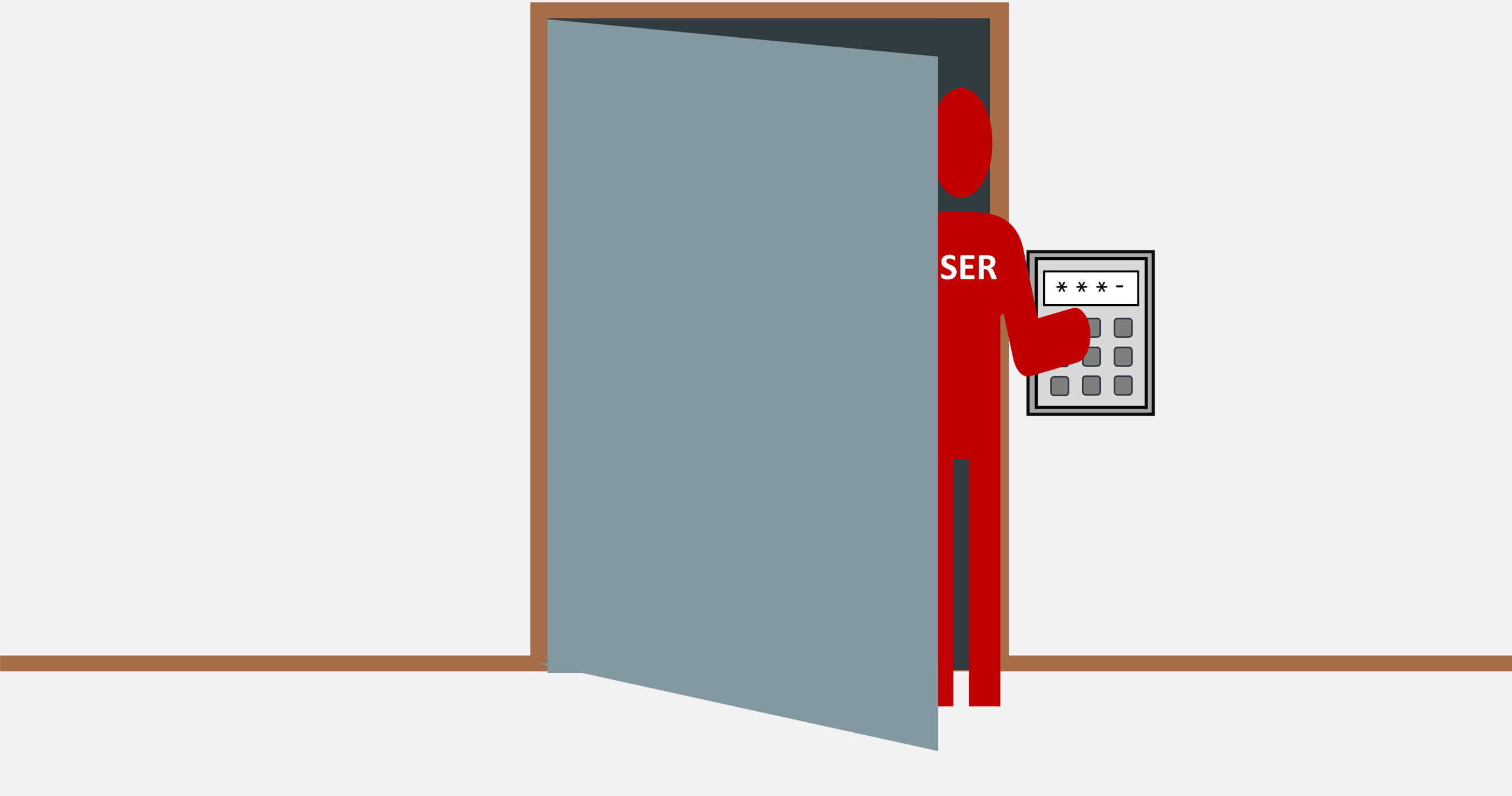
BUSted

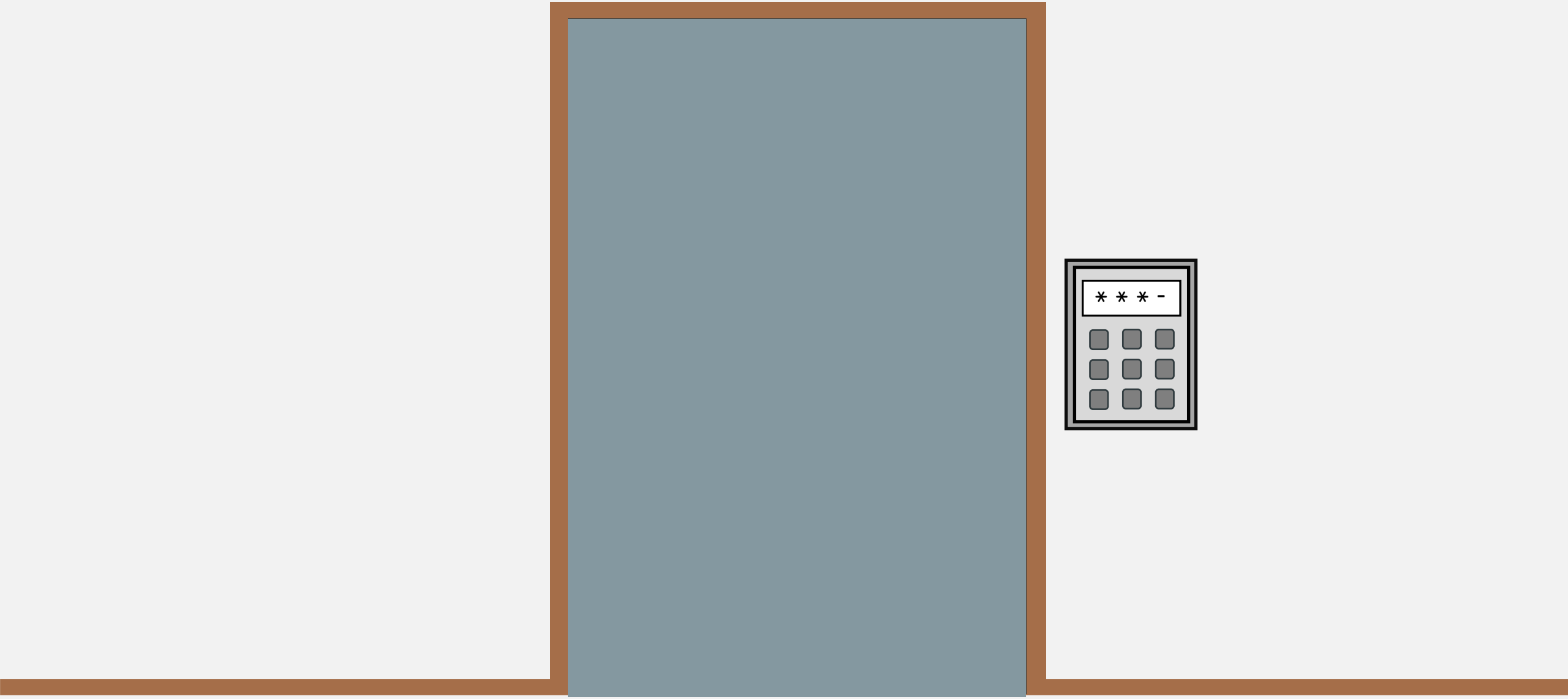
Microarchitectural Side-Channel Attacks on MCUs

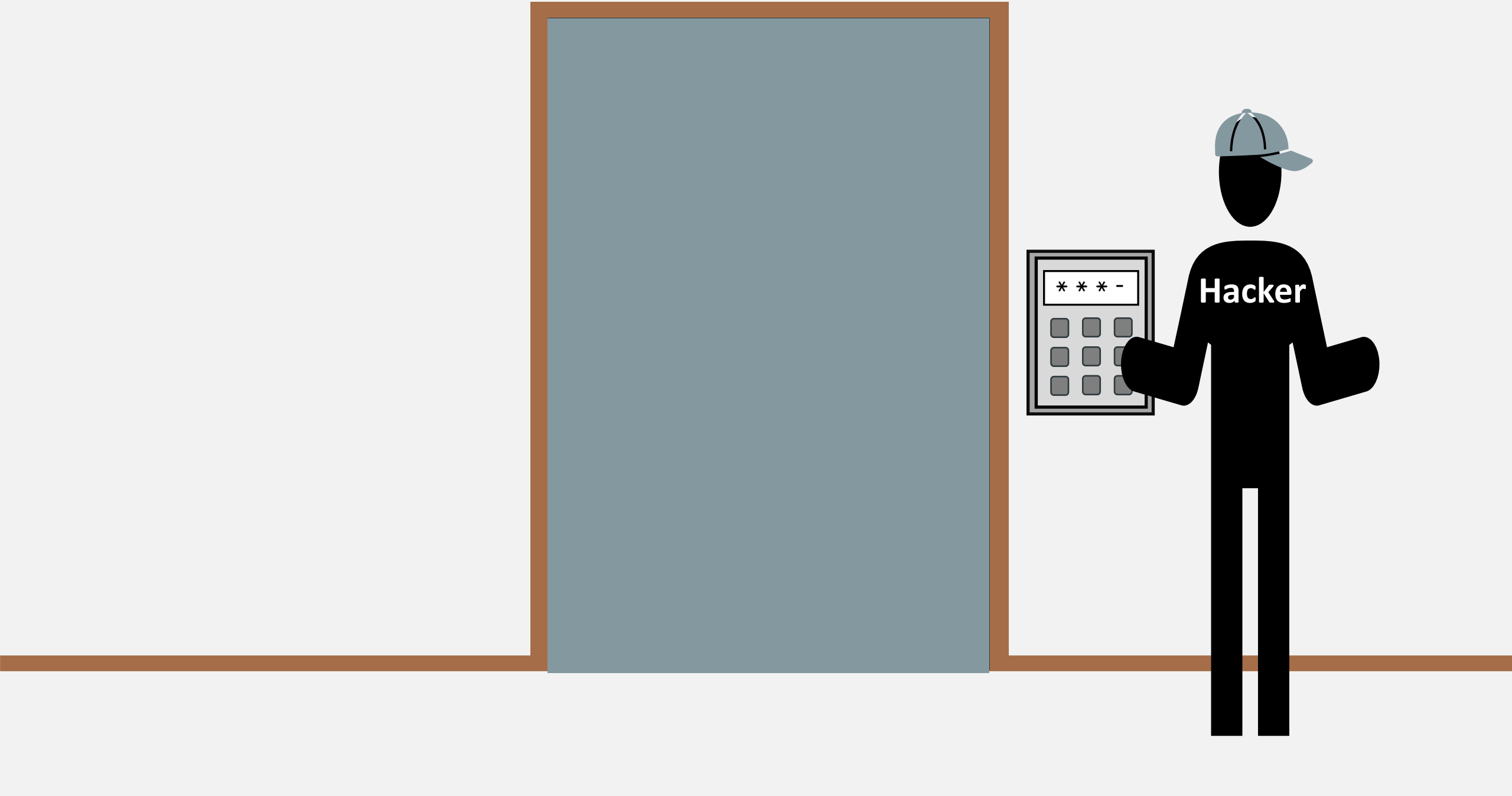


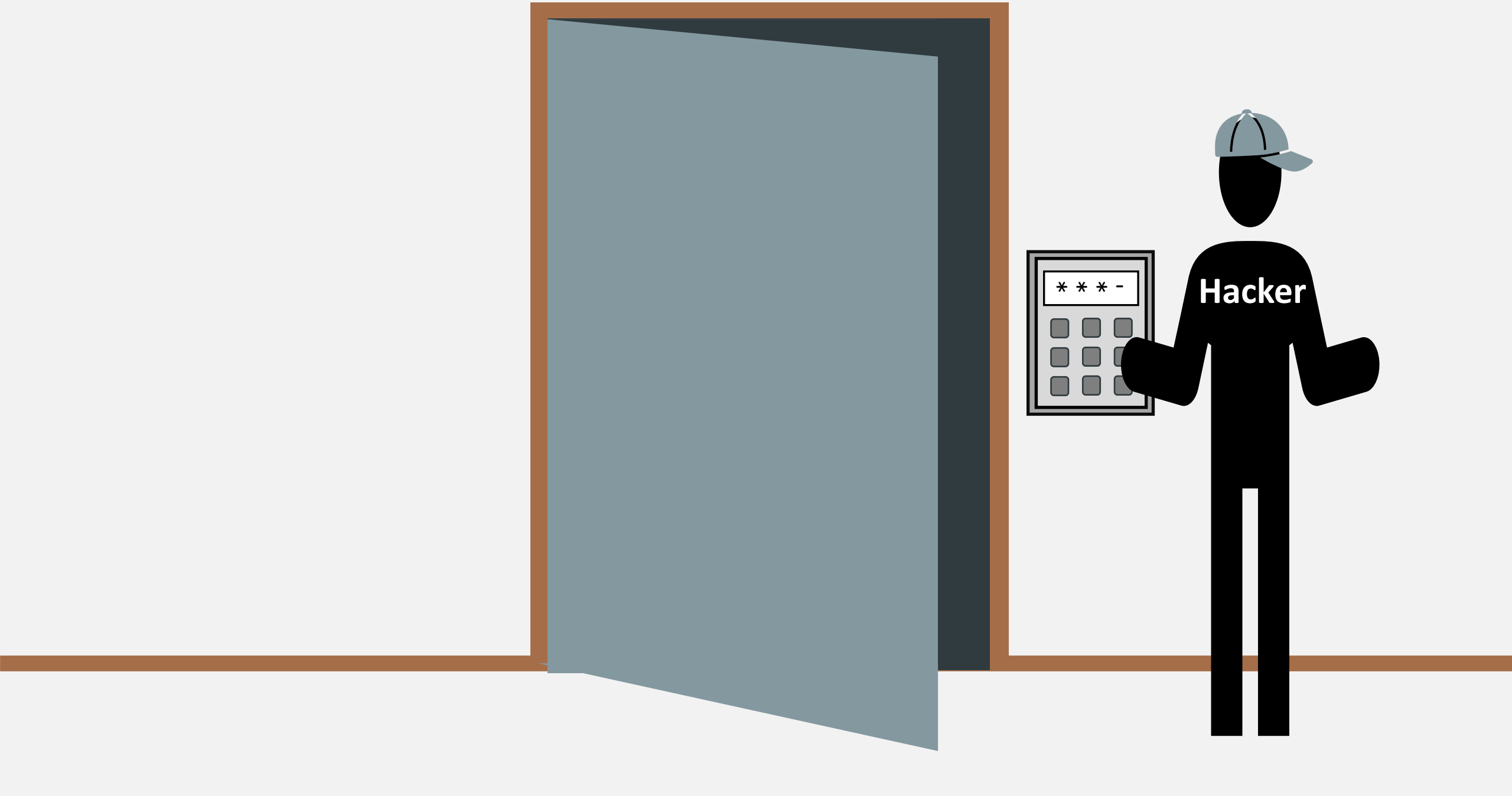

* * * -
USER


* * * -
USER

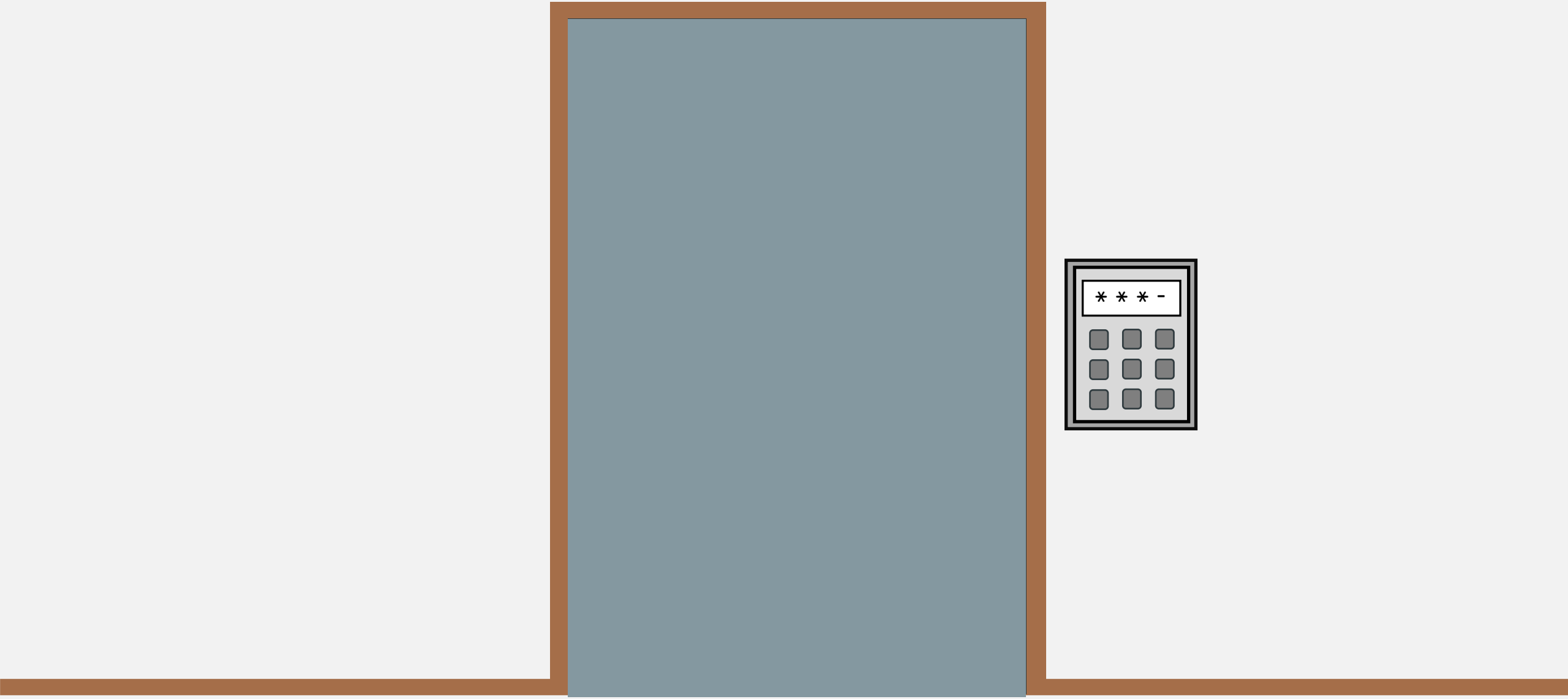

SER
* * * -

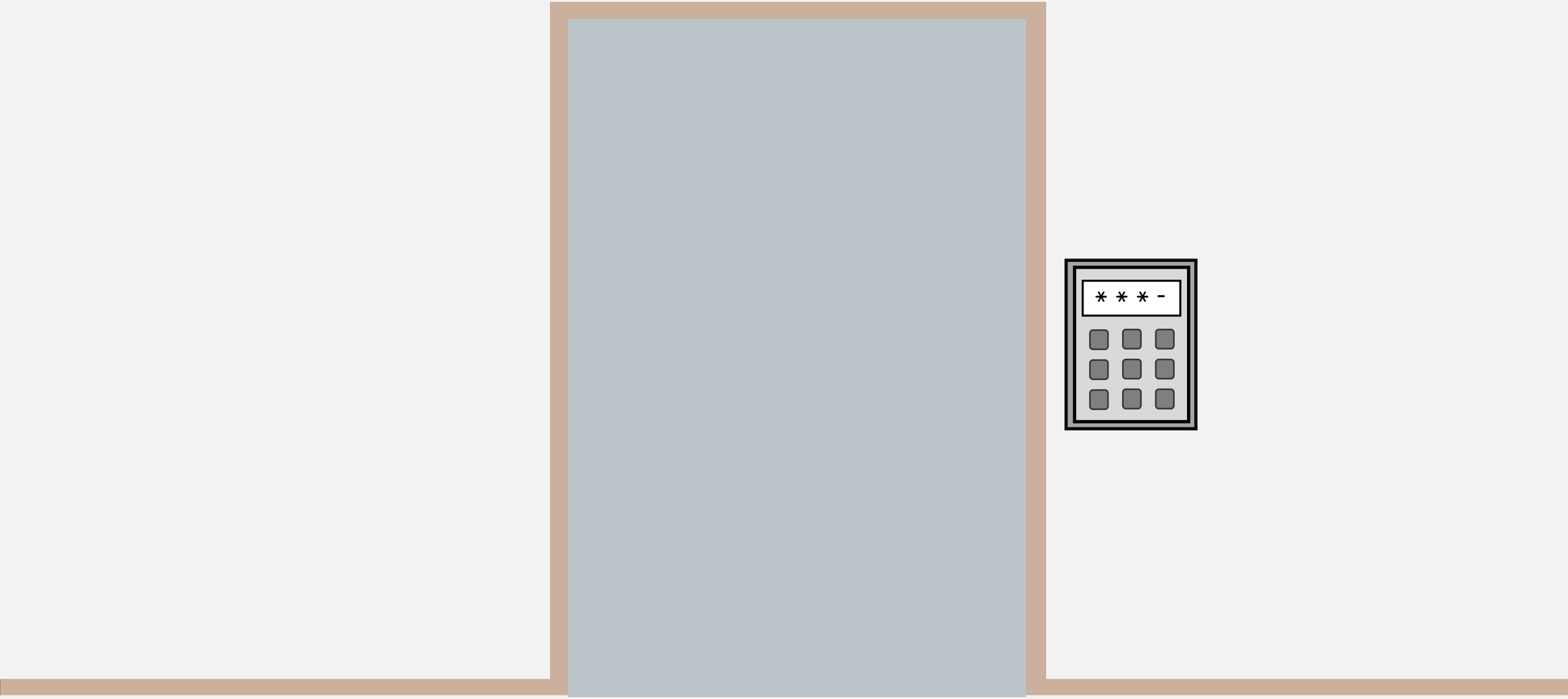



* * * -
Hacker

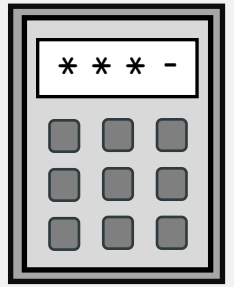

* * * -
Hacker

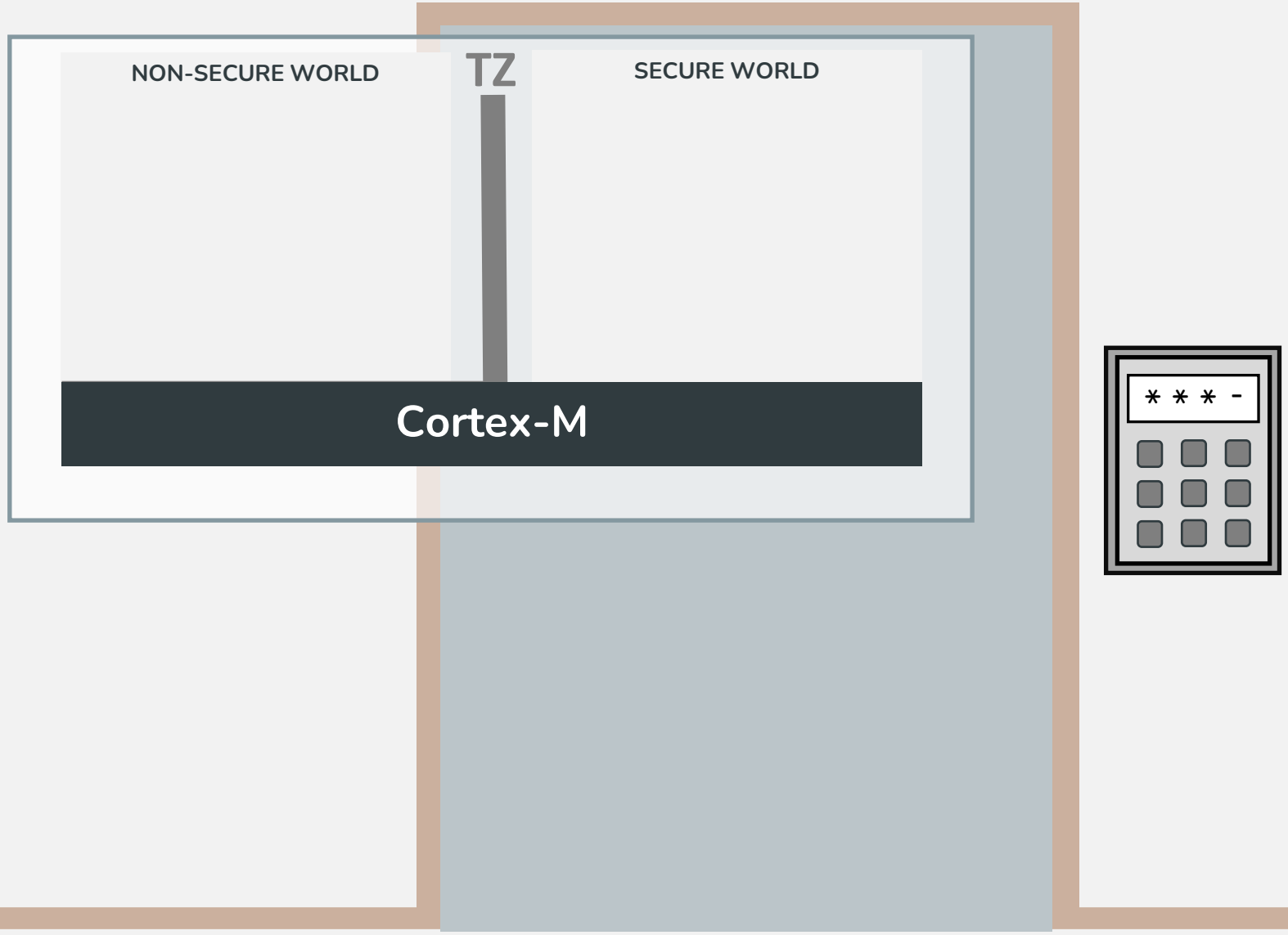

Hacker
* * * -

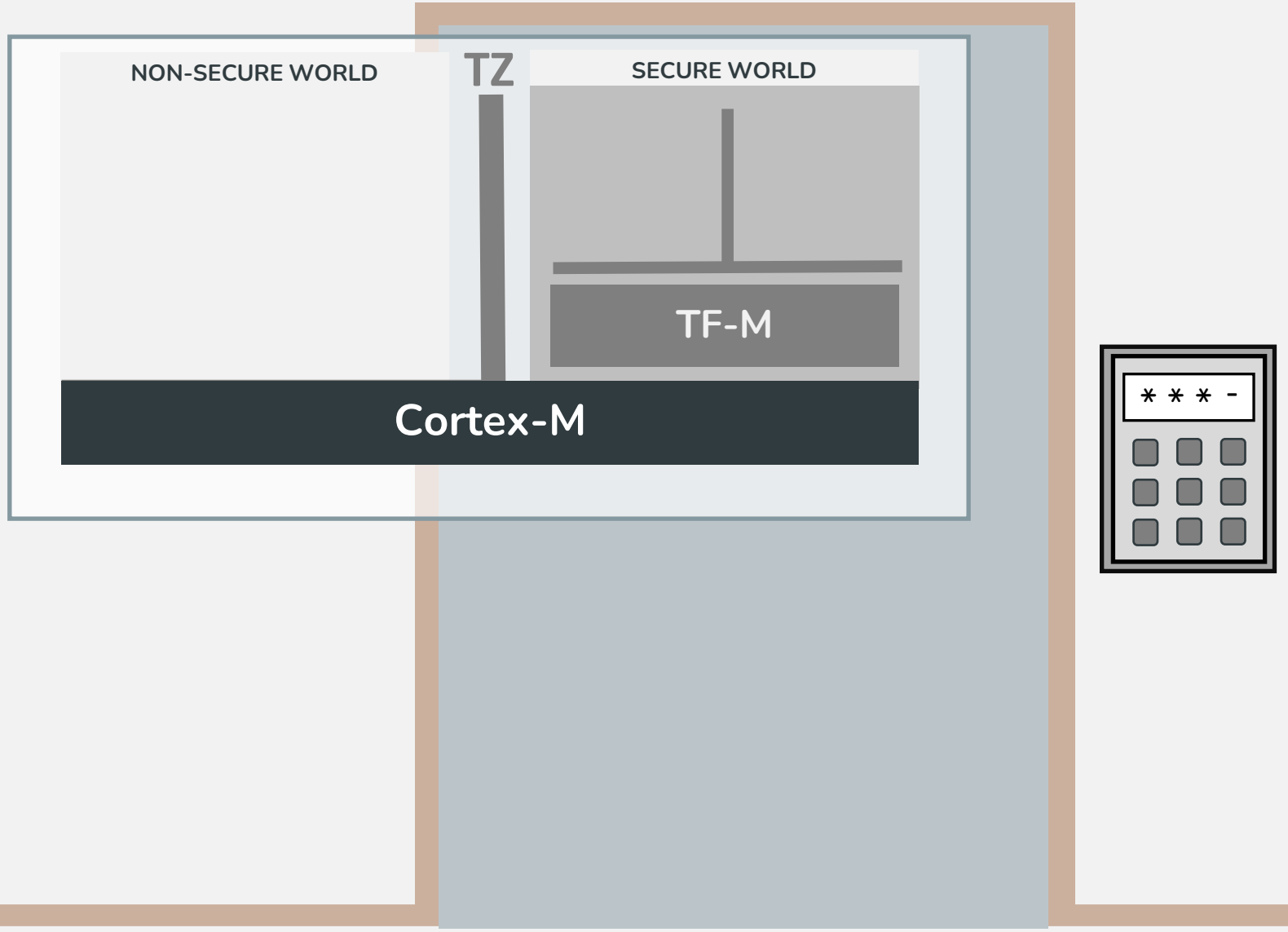


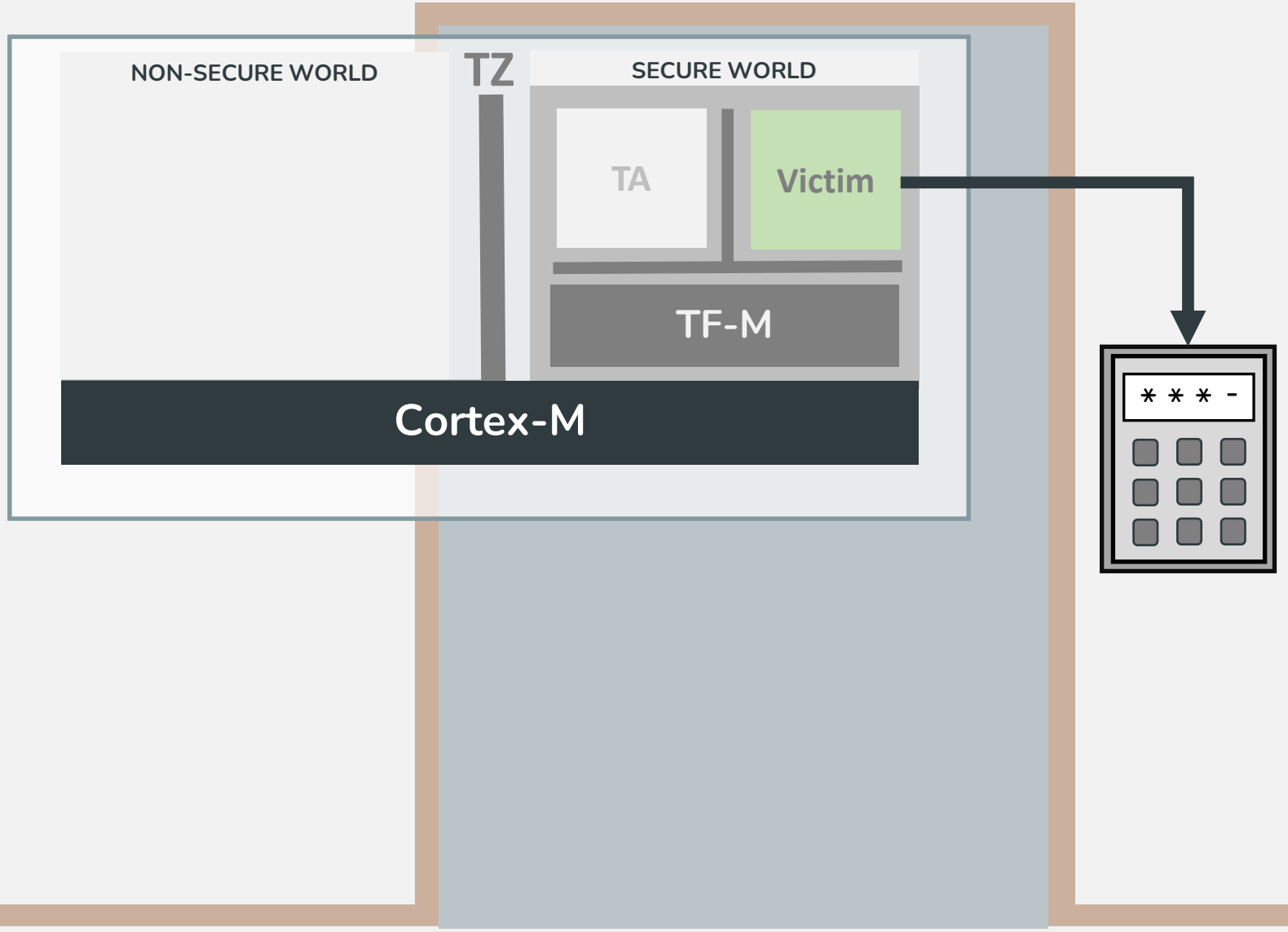


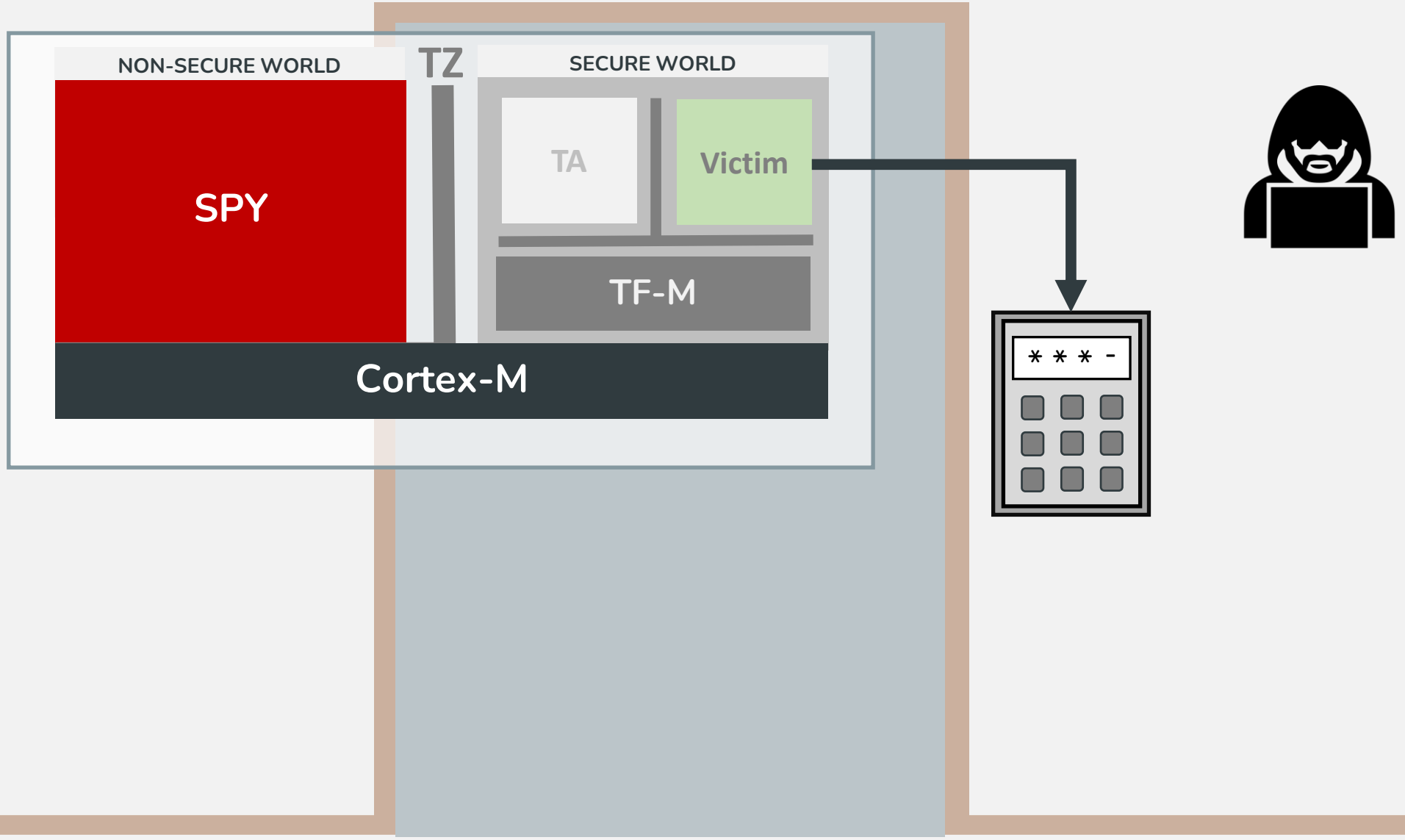
Cortex-M

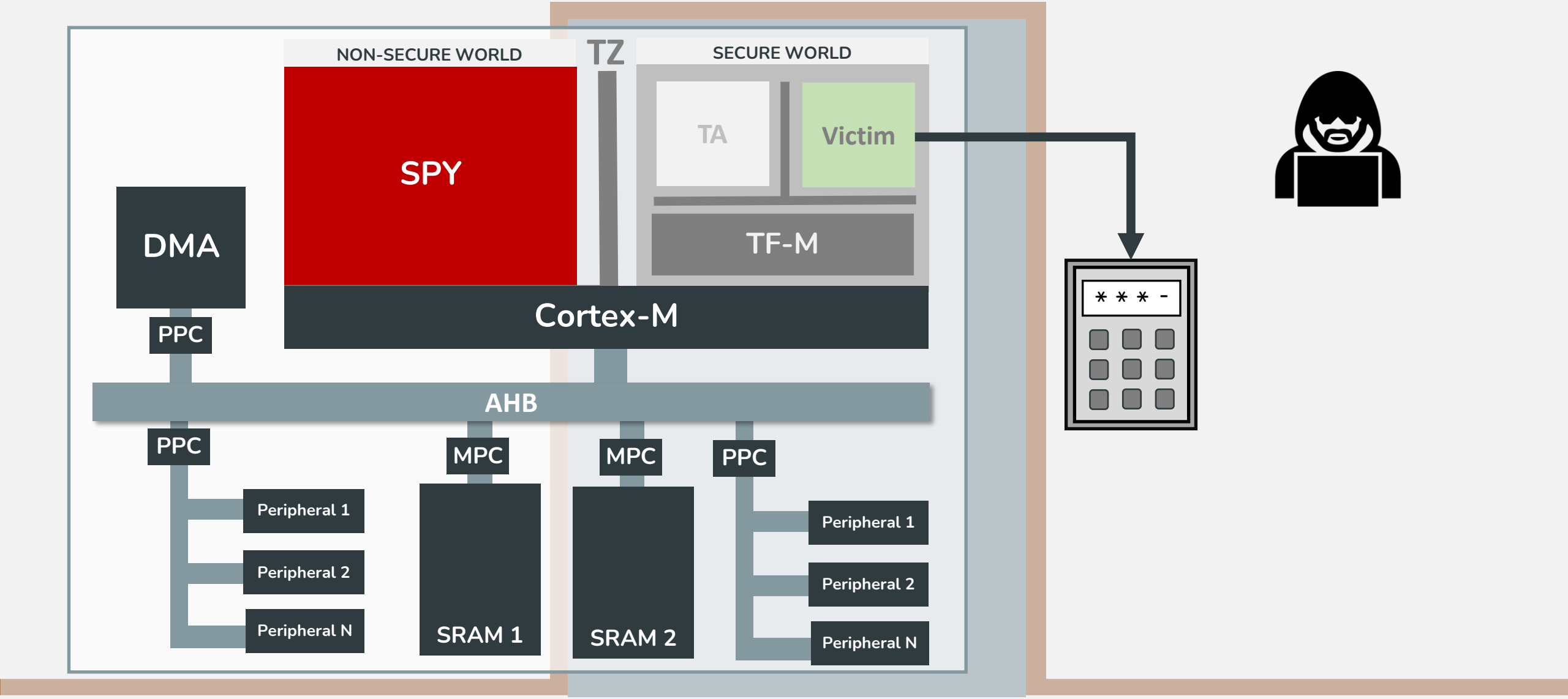


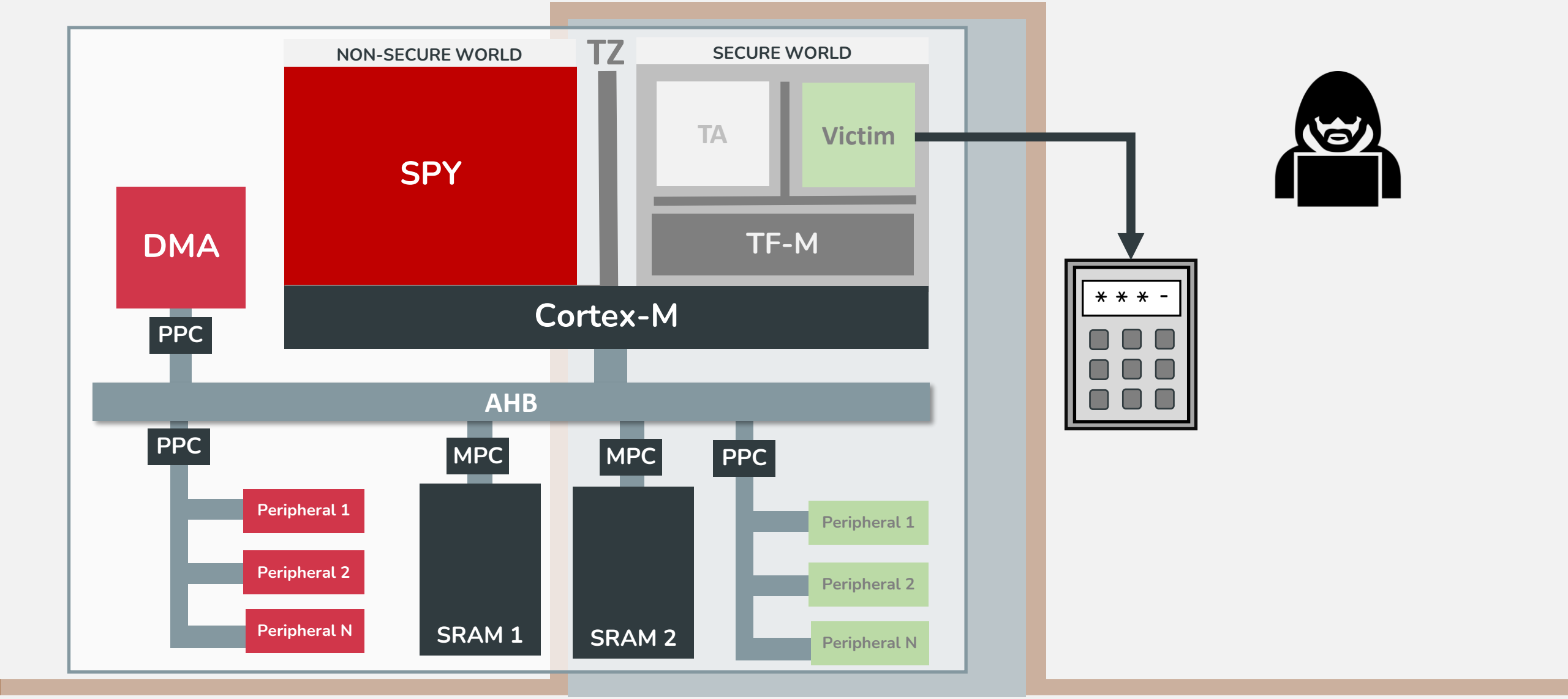

NON-SECURE WORLD
TZ
SECURE WORLD
Cortex-M
* * * -

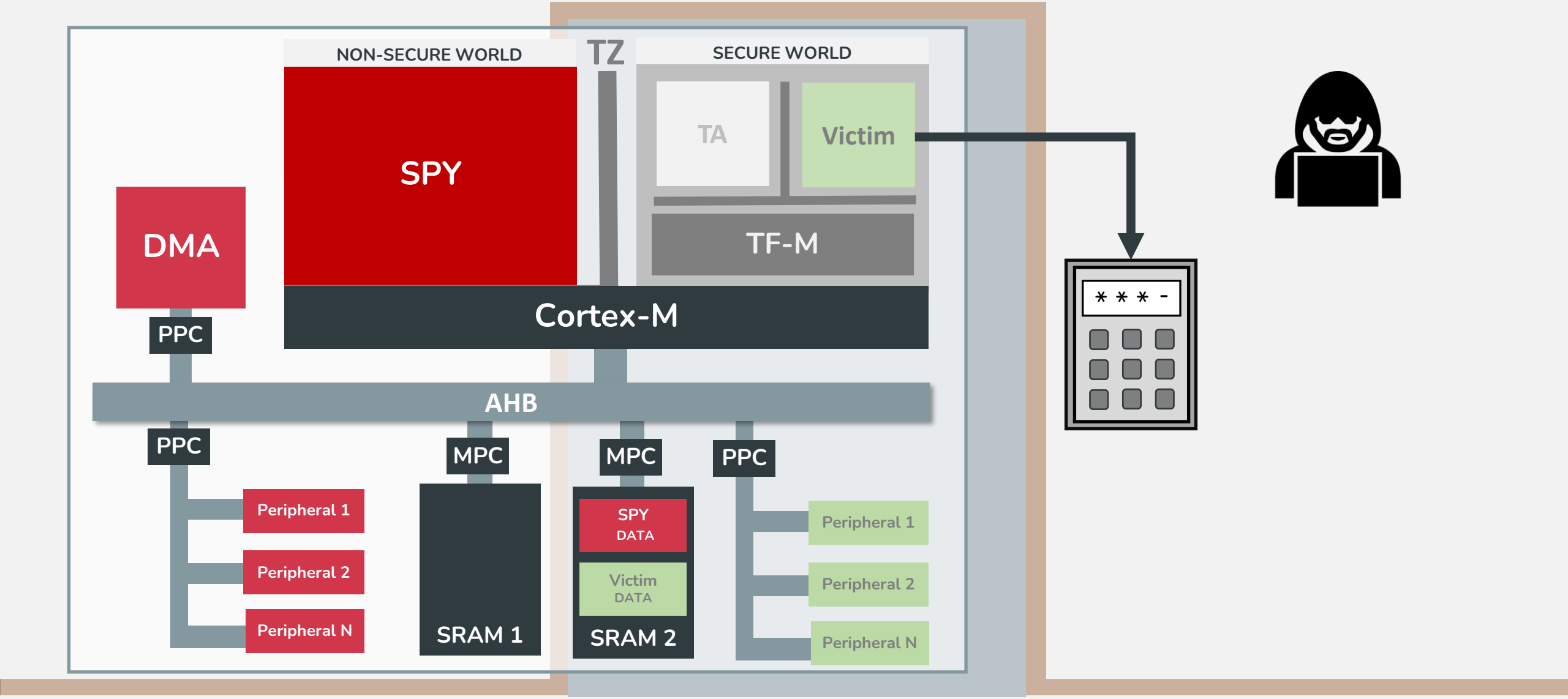

NON-SECURE WORLD
TZ
SECURE WORLD
TF-M
Cortex-M
* * * -


NON-SECURE WORLD
TZ
SECURE WORLD
TA
Victim
TF-M
Cortex-M
* * * -

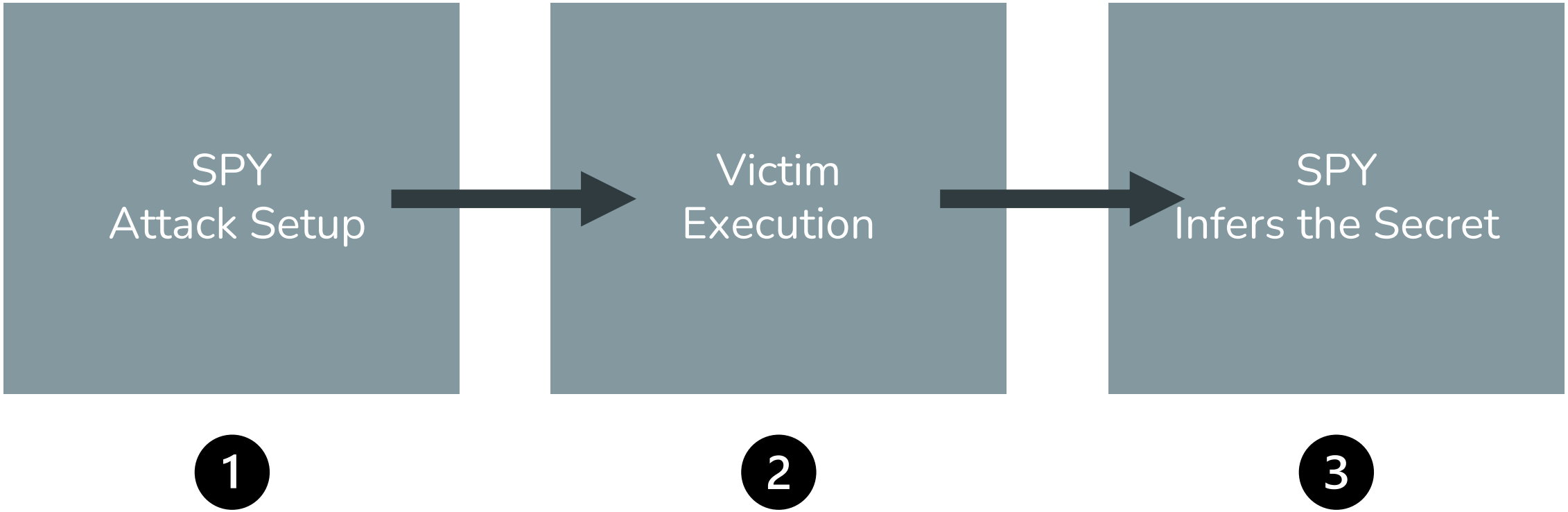

NON-SECURE WORLD
TZ
SECURE WORLD
SPY
TA
Victim
TF-M
Cortex-M
* * * -


NON-SECURE WORLD
TZ
SECURE WORLD
SPY
TA
Victim
TF-M
DMA
Cortex-M
PPC
AHB
PPC
MPC
MPC
PPC
Peripheral 1
Peripheral 2
Peripheral N
SRAM 1
SRAM 2
Peripheral 1
Peripheral 2
Peripheral N
* * * -

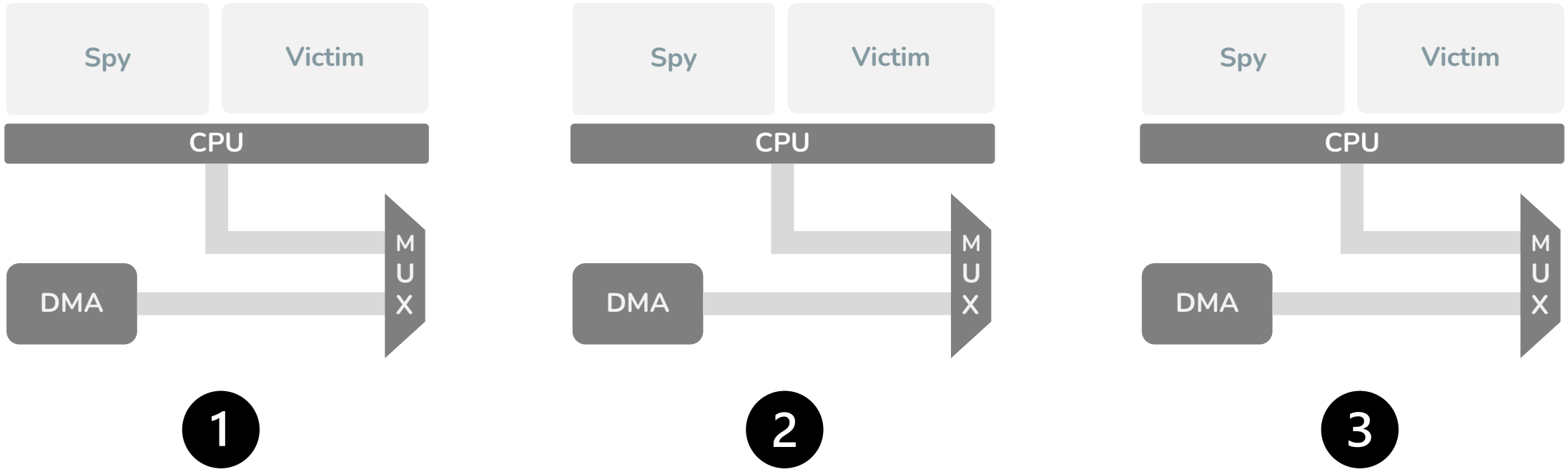

NON-SECURE WORLD
TZ
SECURE WORLD
SPY
TA
Victim
TF-M
DMA
Cortex-M
PPC
AHB
PPC
MPC
MPC
PPC
Peripheral 1
Peripheral 2
Peripheral N
SRAM 1
SRAM 2
Peripheral 1
Peripheral 2
Peripheral N
* * * -


NON-SECURE WORLD
TZ
SECURE WORLD
SPY
TA
Victim
TF-M
DMA
Cortex-M
PPC
AHB
PPC
MPC
MPC
PPC
Peripheral 1
Peripheral 2
Peripheral N
SRAM 1
SPY
DATA
Victim
DATA
SRAM 2
Peripheral 1
Peripheral 2
Peripheral N
* * * -

Attack Phases

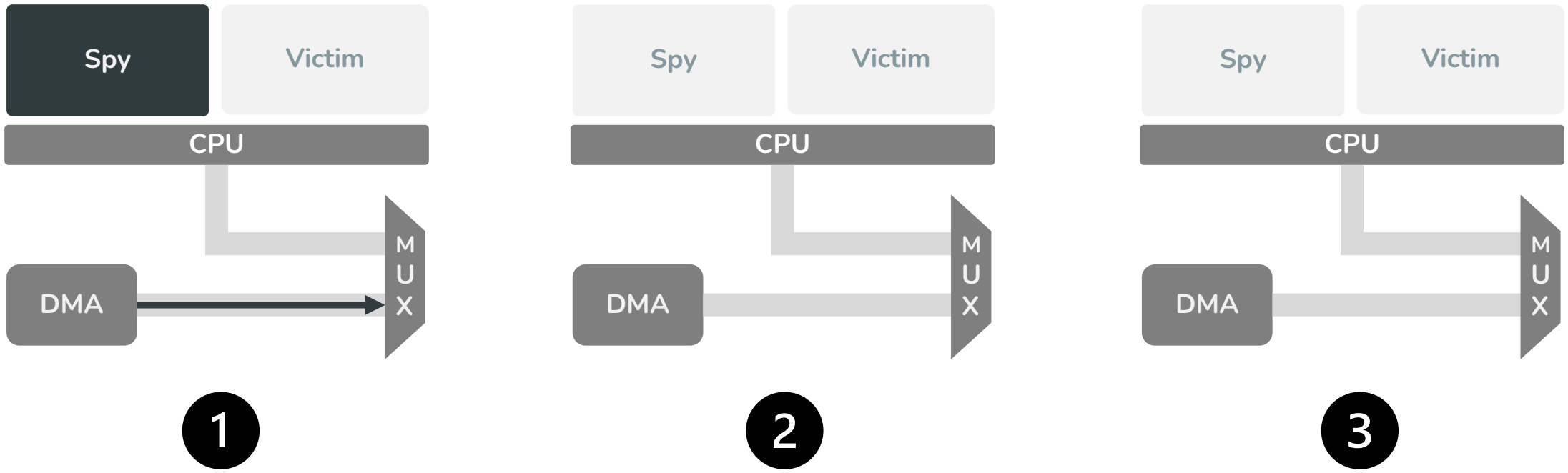


Challenges to mount BUSted

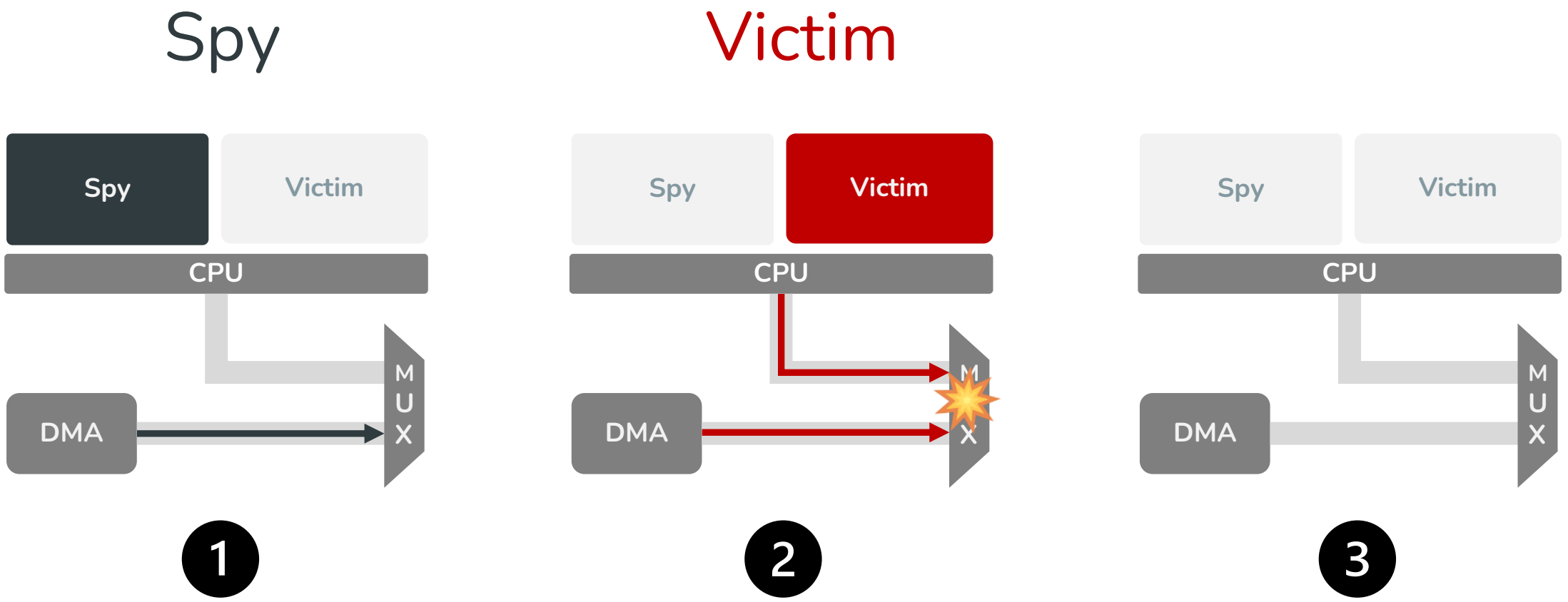


Challenges to mount BUSted

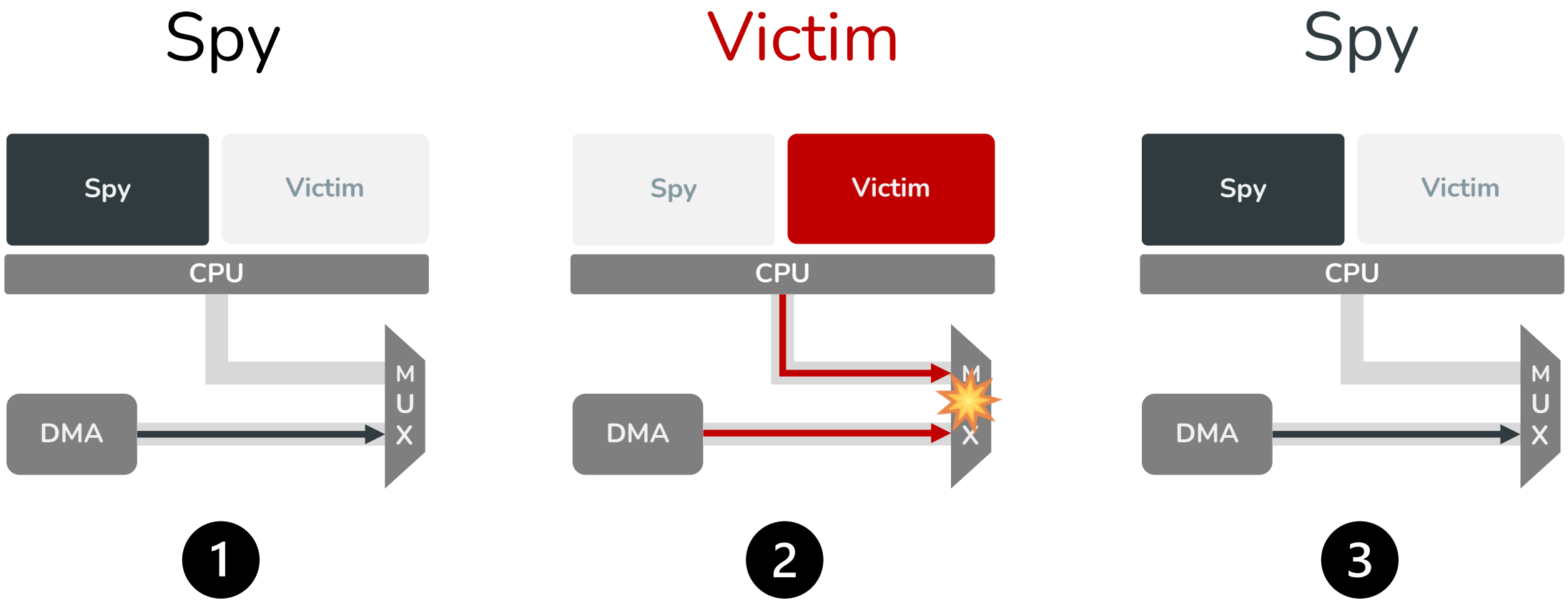
Spy



Challenges to mount BUSted

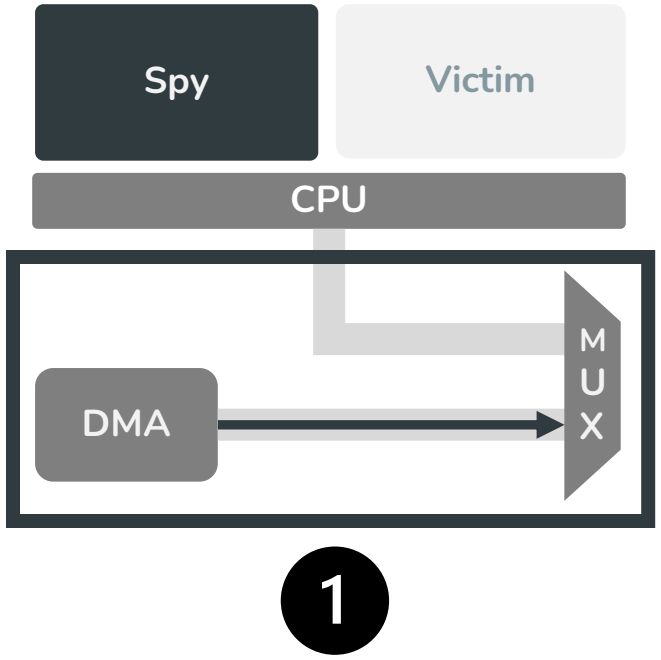


Challenges to mount BUSted



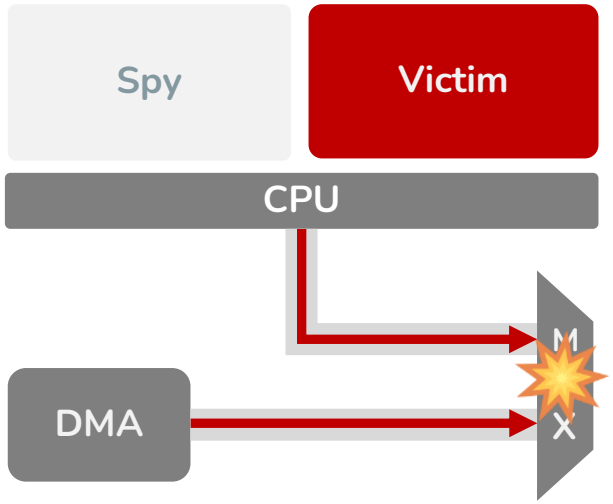
Challenges to mount BUSted

Spy



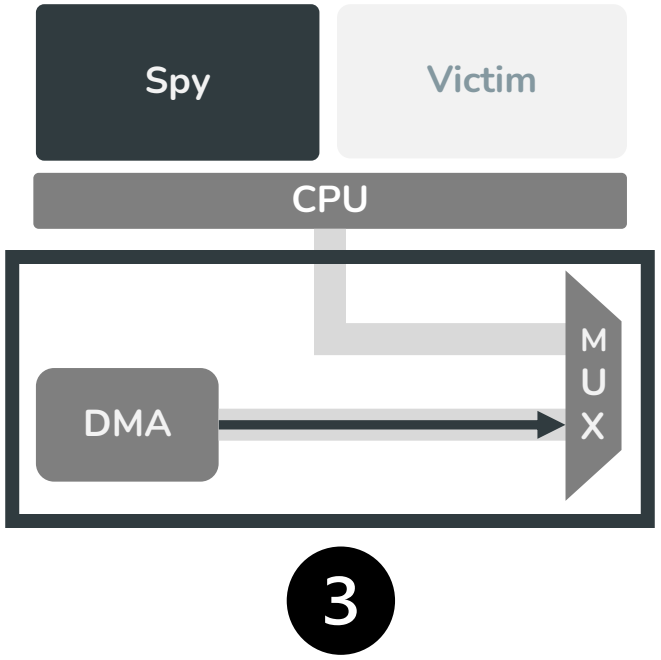
1

Victim



2

Spy



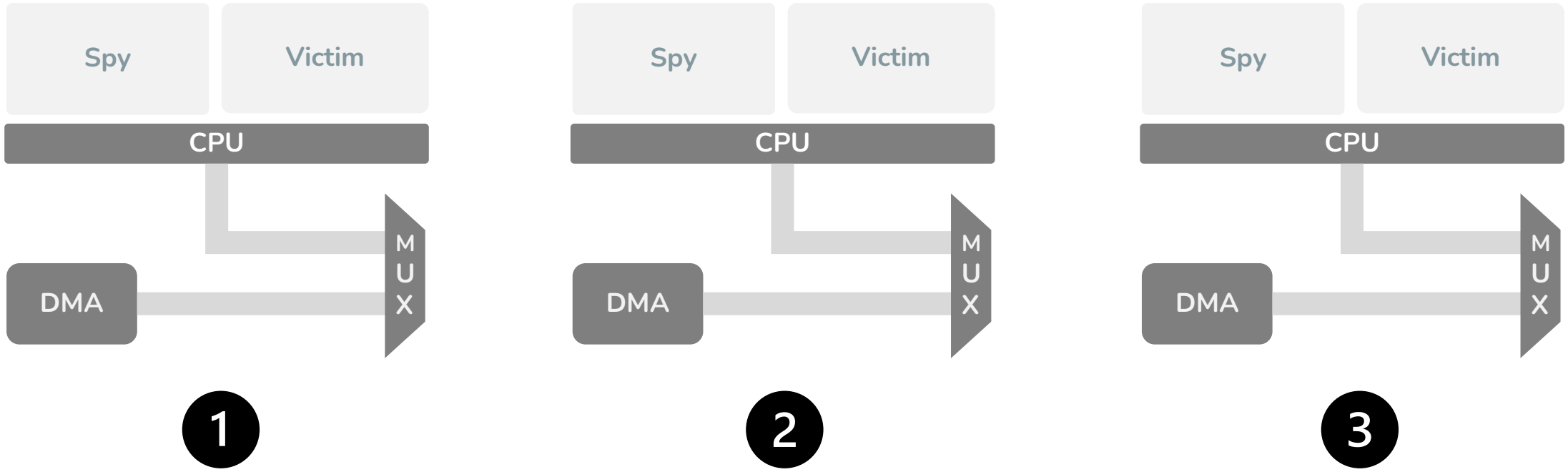
3

No Difference

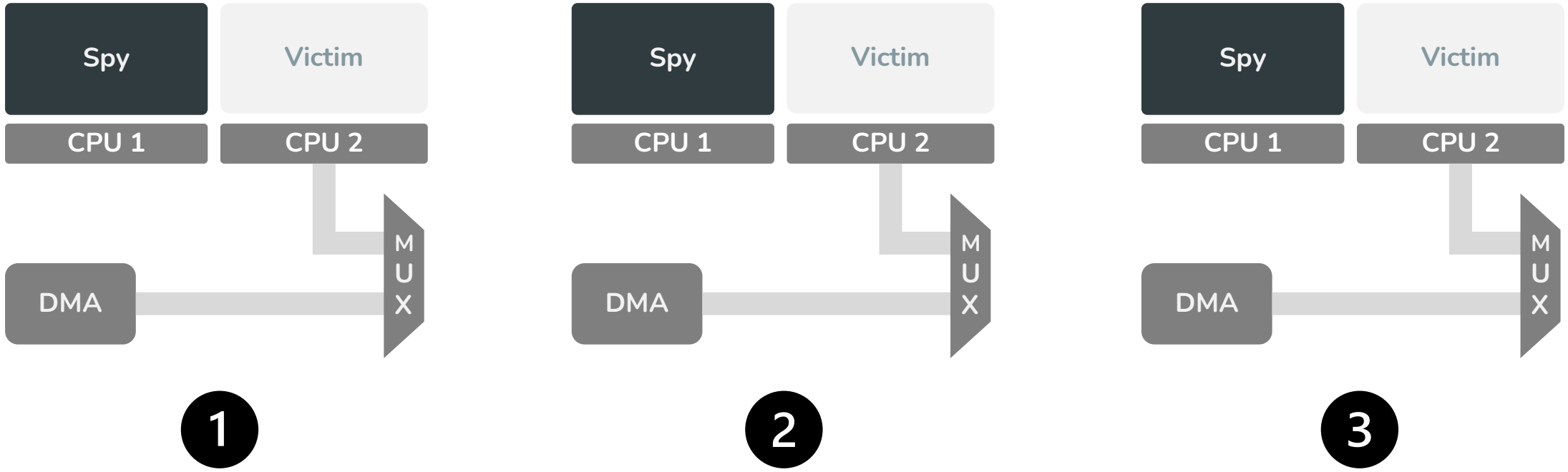
Challenges to mount BUSted

C1 - The bus is a stateless component;

Challenges to mount BUSted

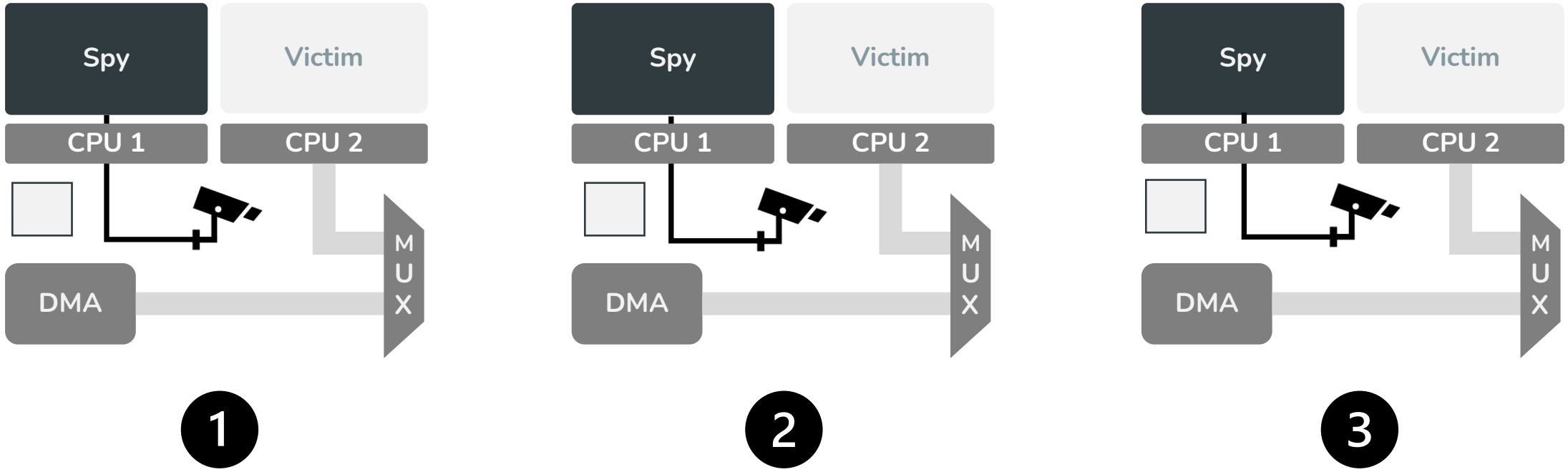


Challenges to mount BUSted



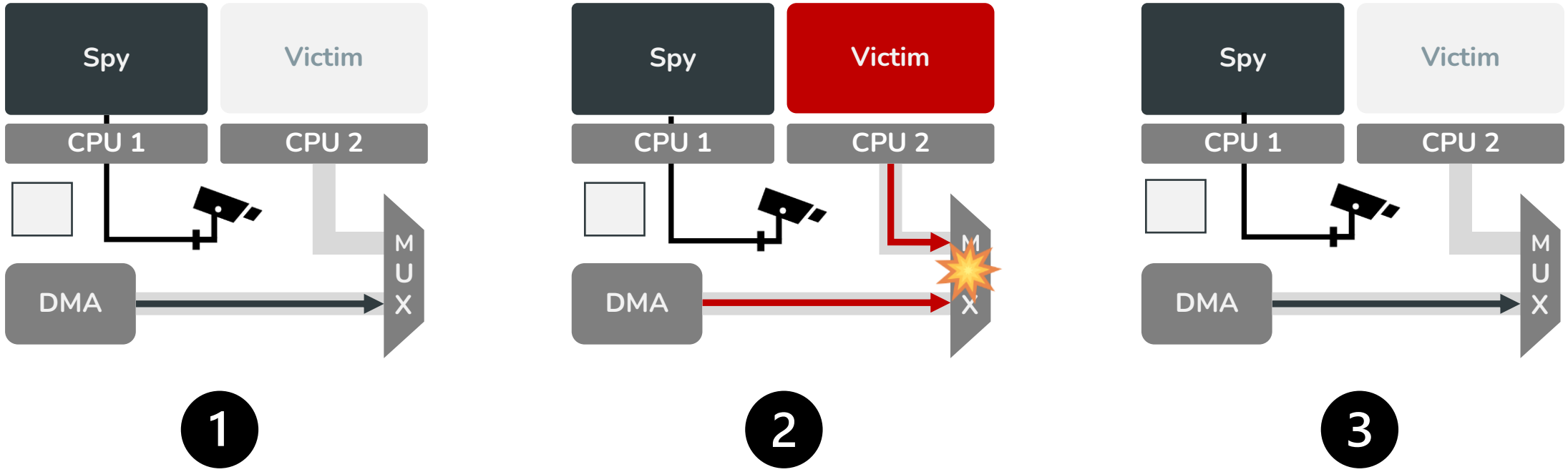
Dual-Core

Challenges to mount BUSted



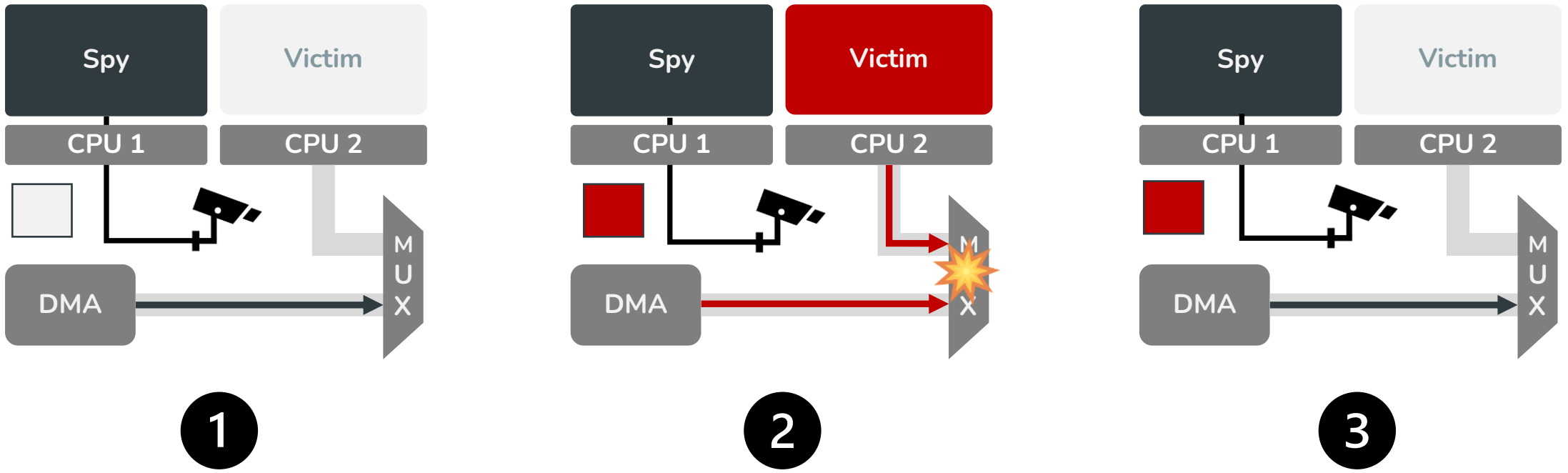
Always Spying

Challenges to mount BUSted



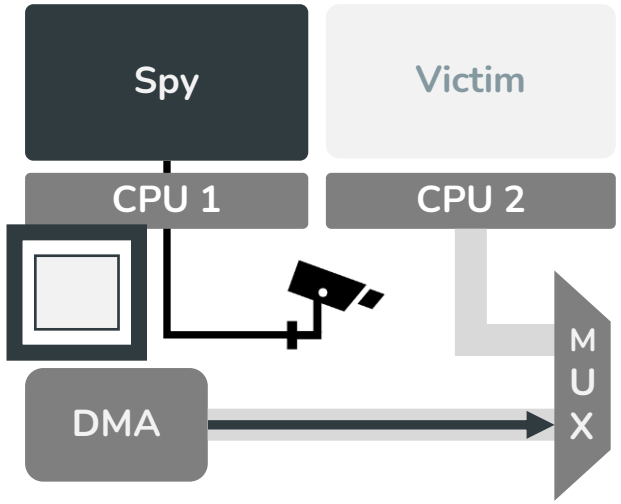
Always Spying

Challenges to mount BUSted

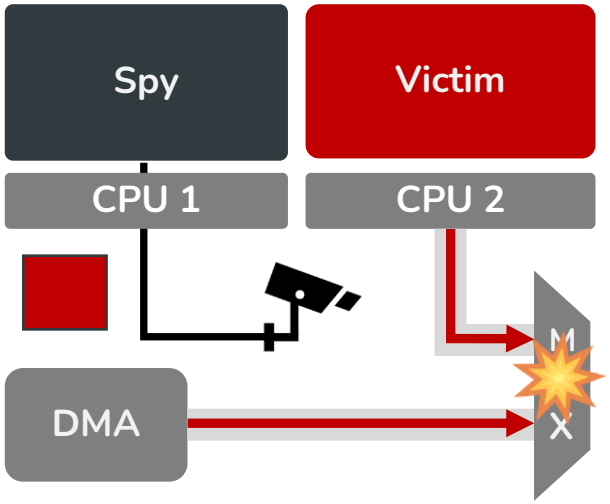


Snapshot of the Bus Contention

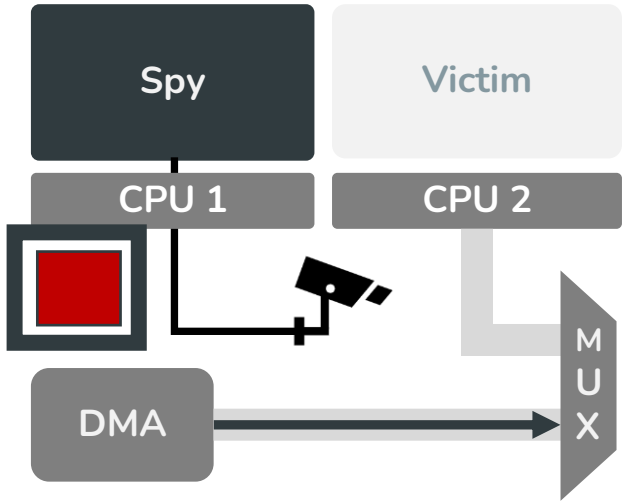
Challenges to mount BUSted



1



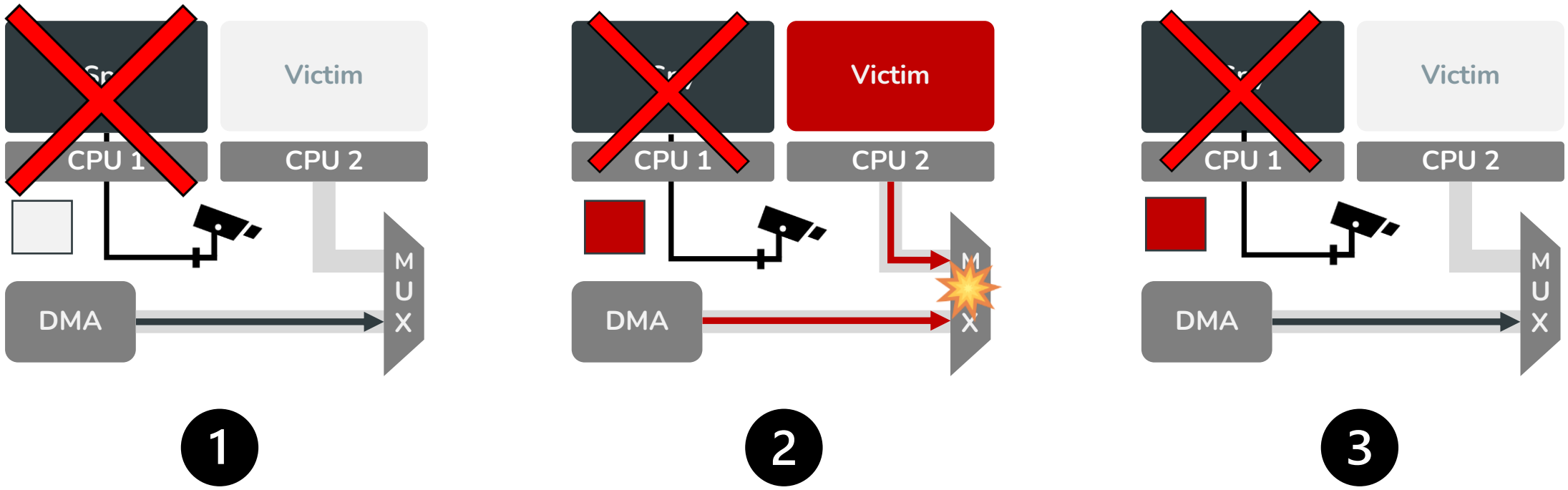
2



3

Different States

Challenges to mount BUSted



Single-Core MCU!!!

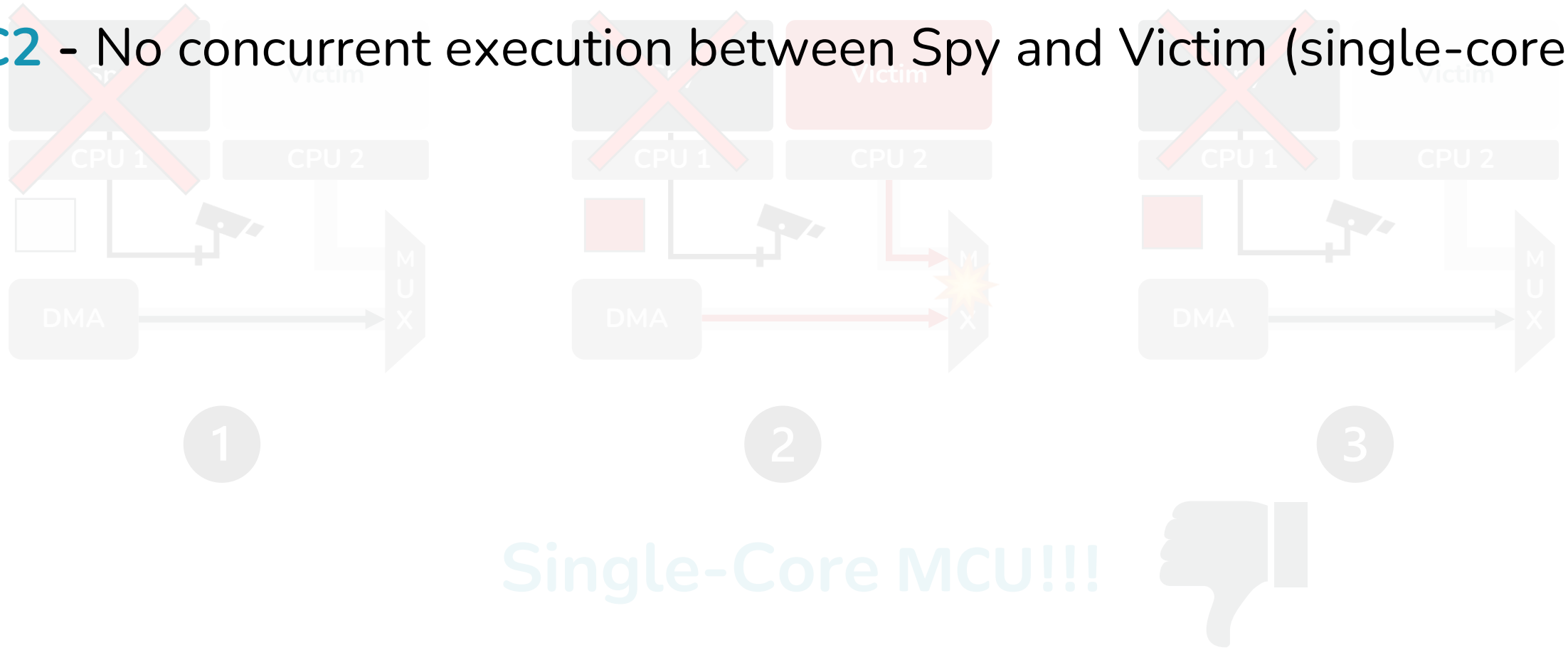
Challenges to mount BUSted

C1 - The bus is a stateless component;

Challenges to mount BUSted

C1 - The bus is a stateless component;

C2 - No concurrent execution between Spy and Victim (single-core);

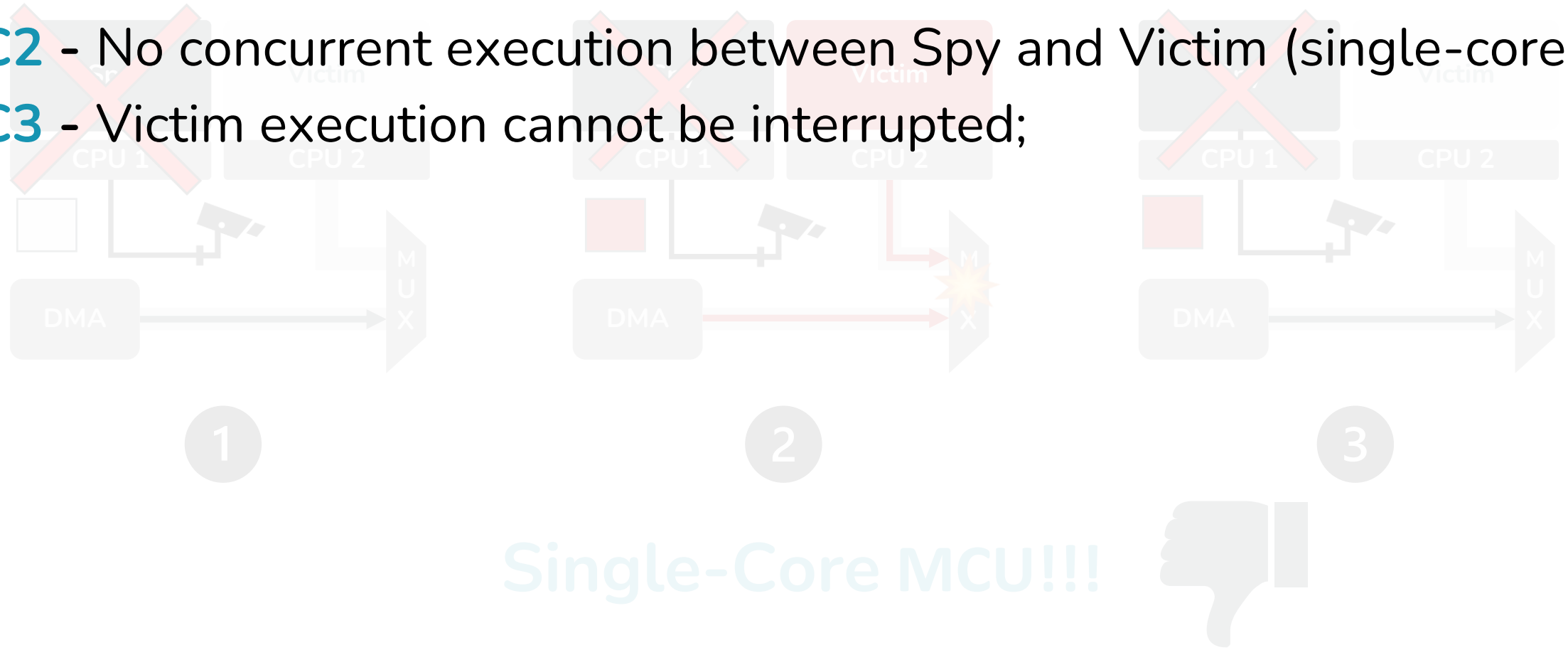


Challenges to mount BUSted

C1 - The bus is a stateless component;

C2 - No concurrent execution between Spy and Victim (single-core);

C3 - Victim execution cannot be interrupted;



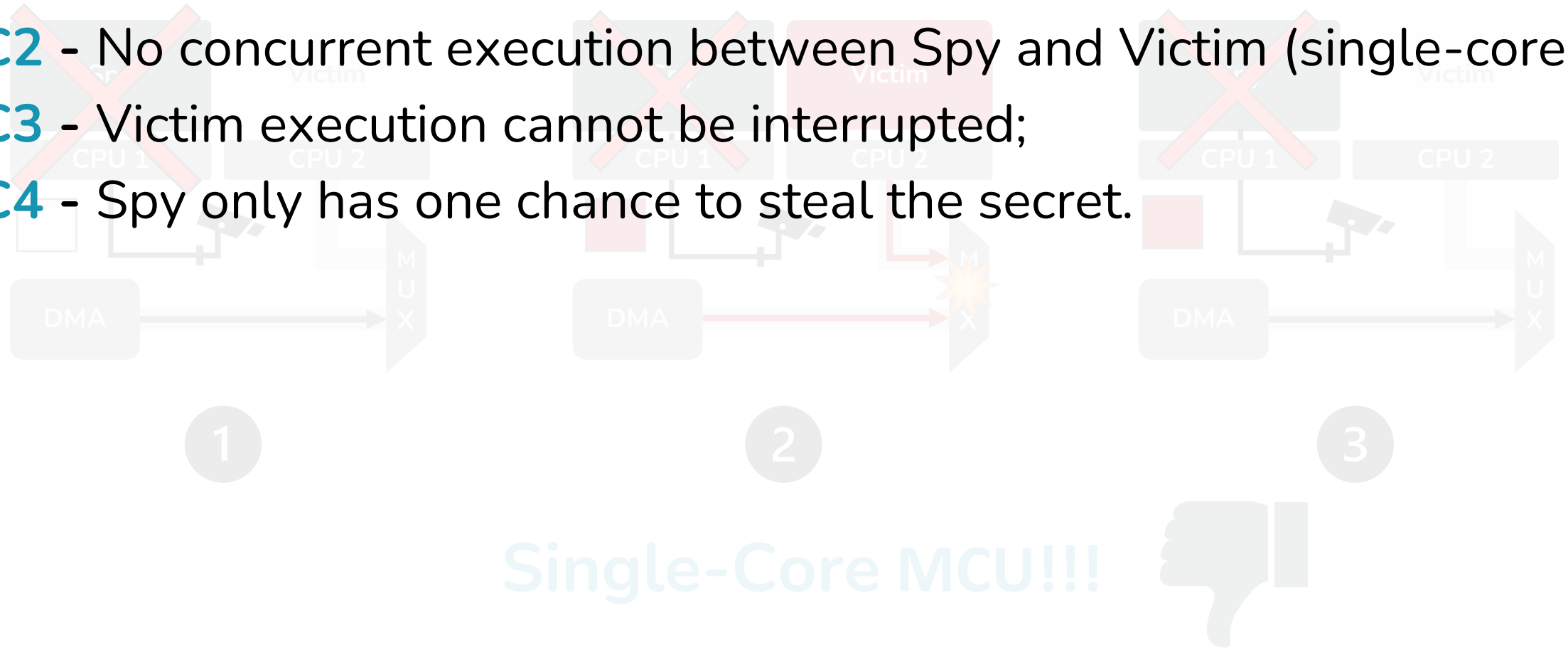
Challenges to mount BUSted

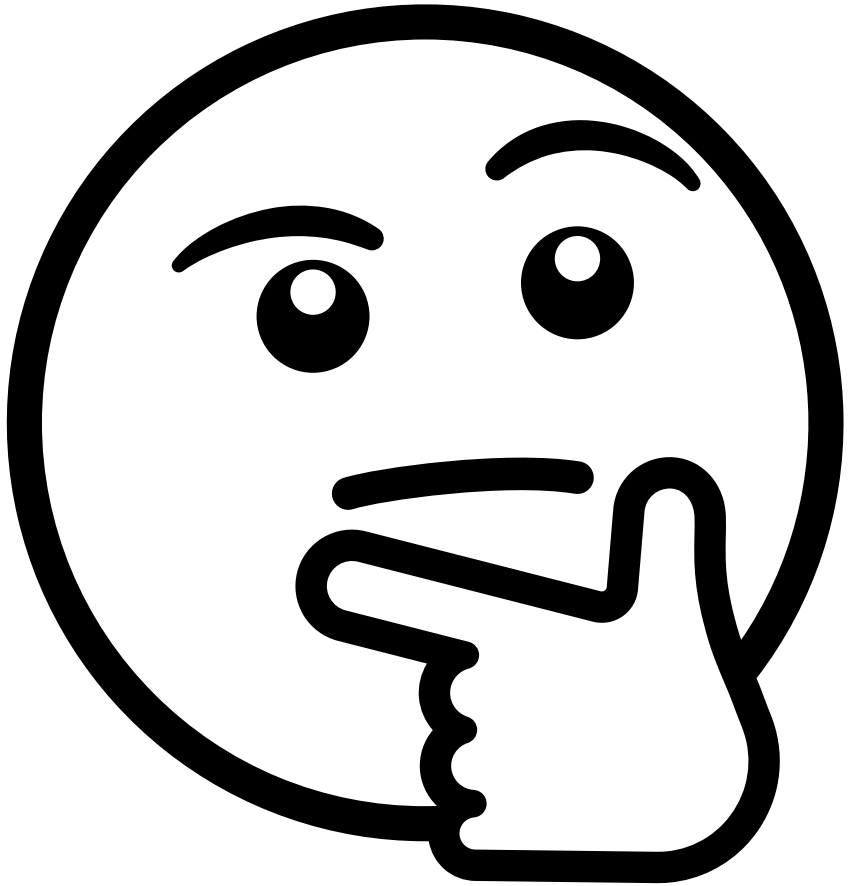
C1 - The bus is a stateless component;

C2 - No concurrent execution between Spy and Victim (single-core);

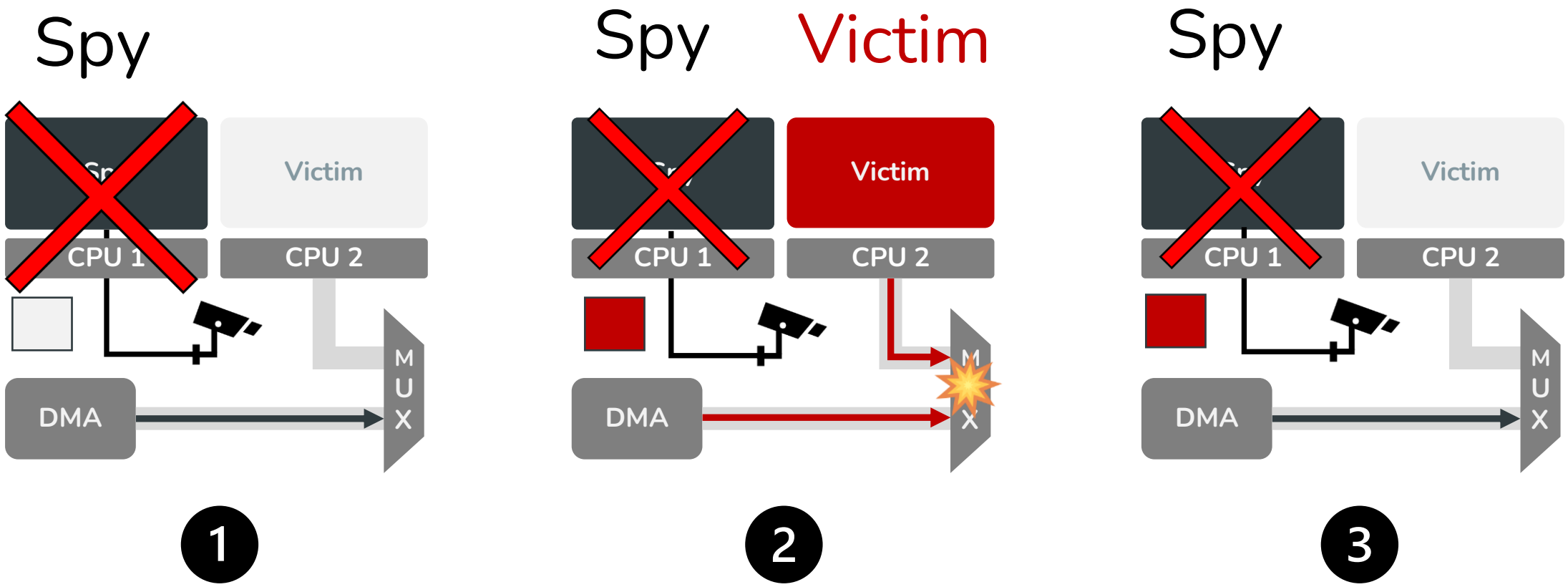
C3 - Victim execution cannot be interrupted;

C4 - Spy only has one chance to steal the secret.



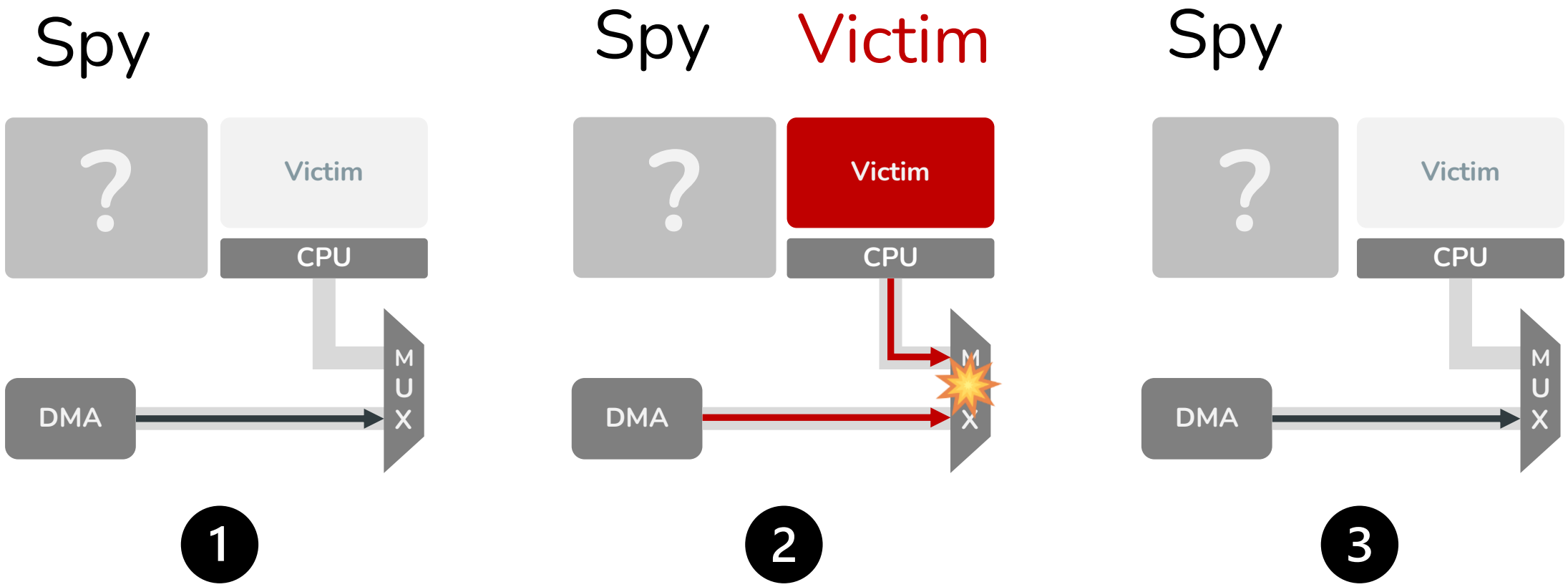


BUSted Solution



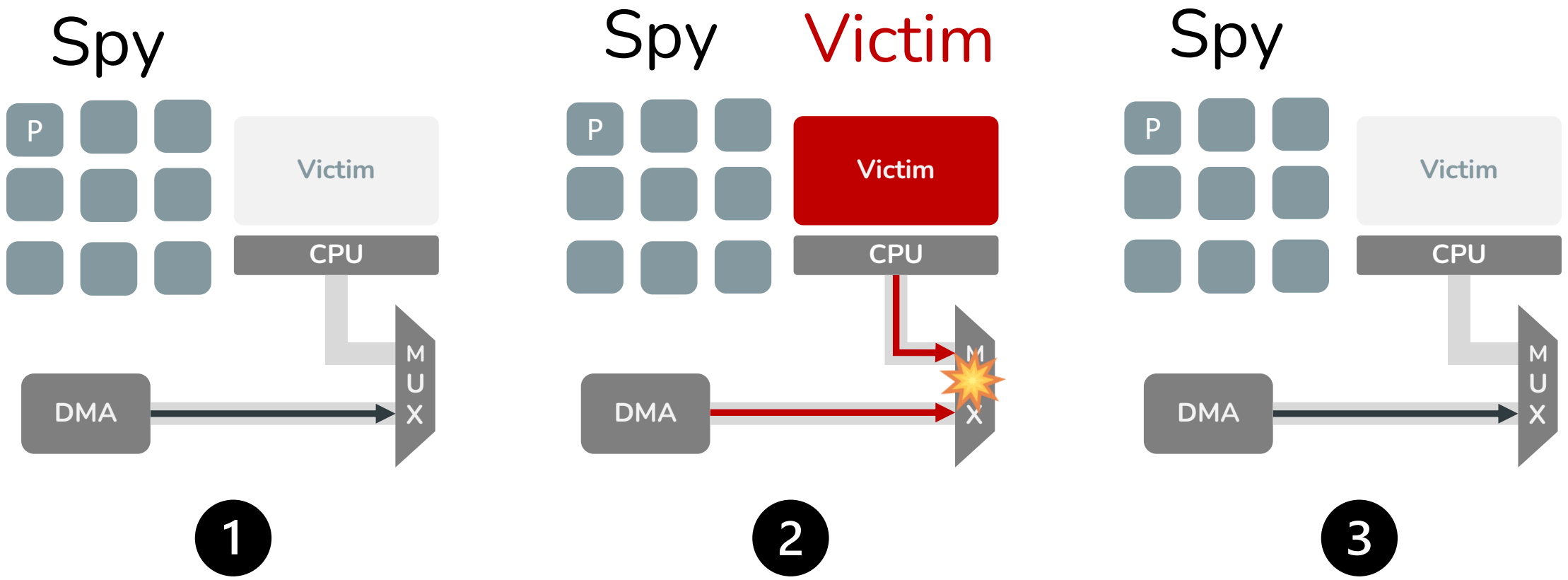
Single-Core MCU!!!

BUSted Solution



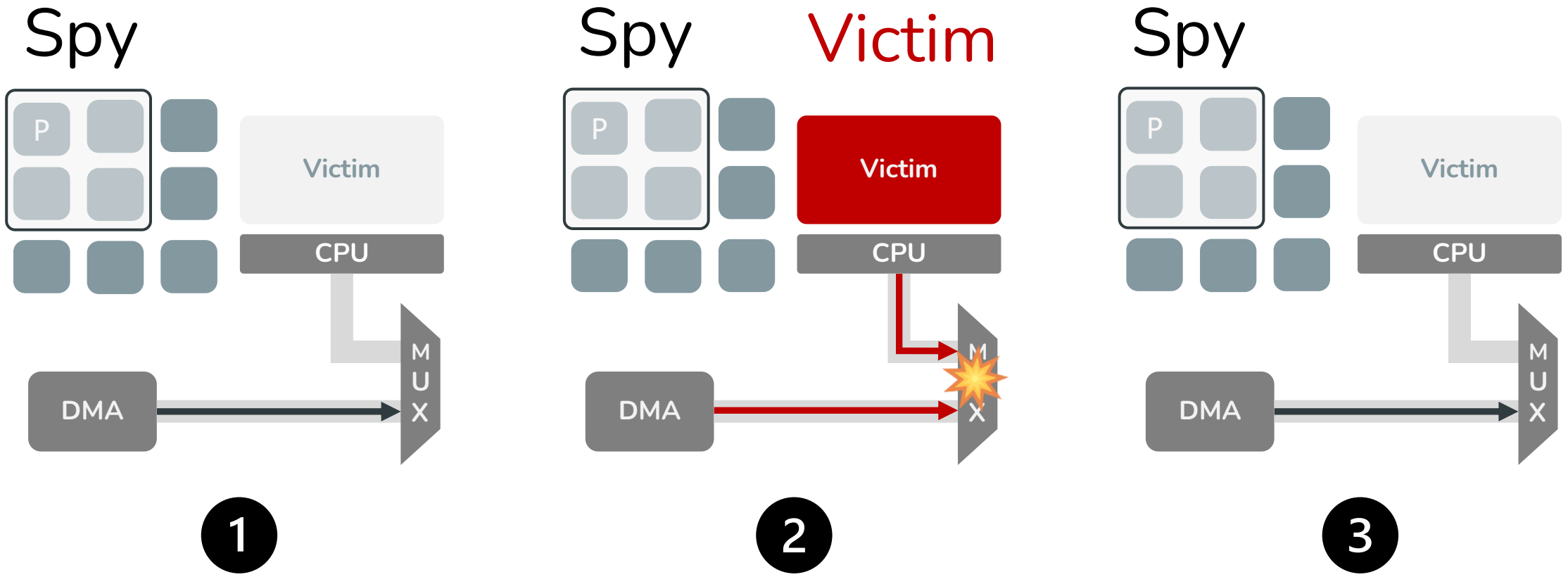
Single-Core MCU!!!

BUSted Solution



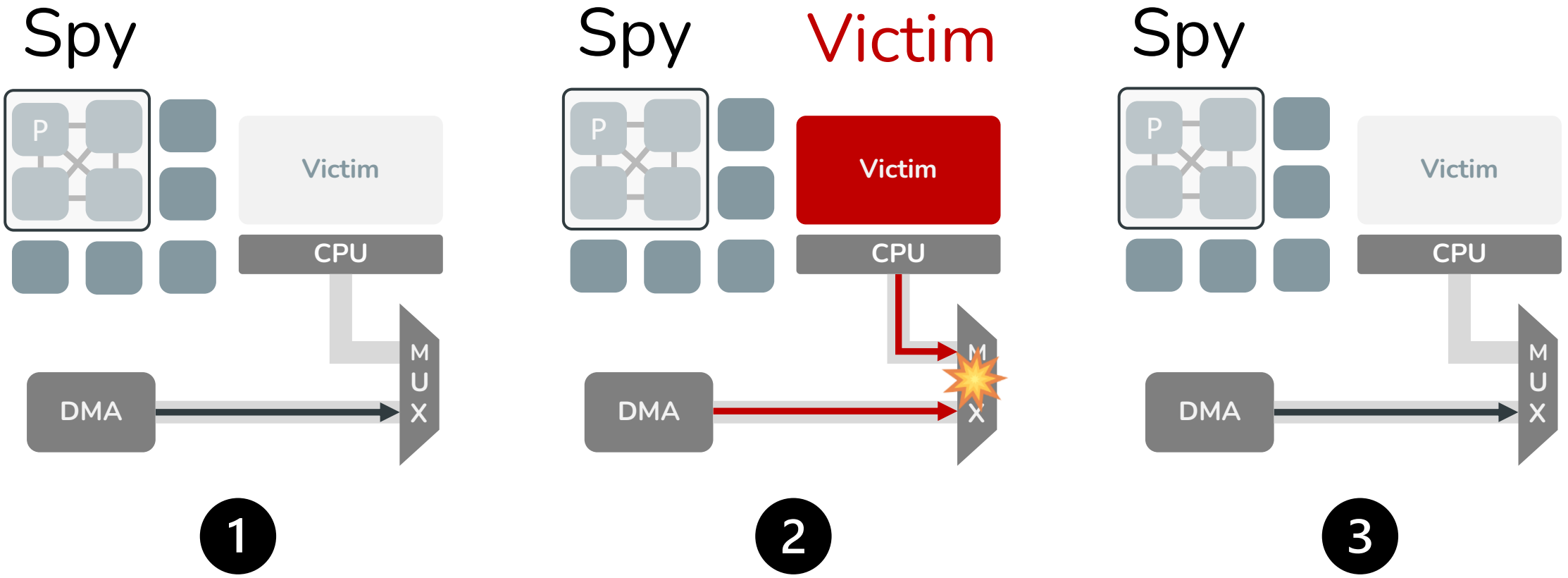
MCUs have a lot of Peripherals!!

BUSted Solution



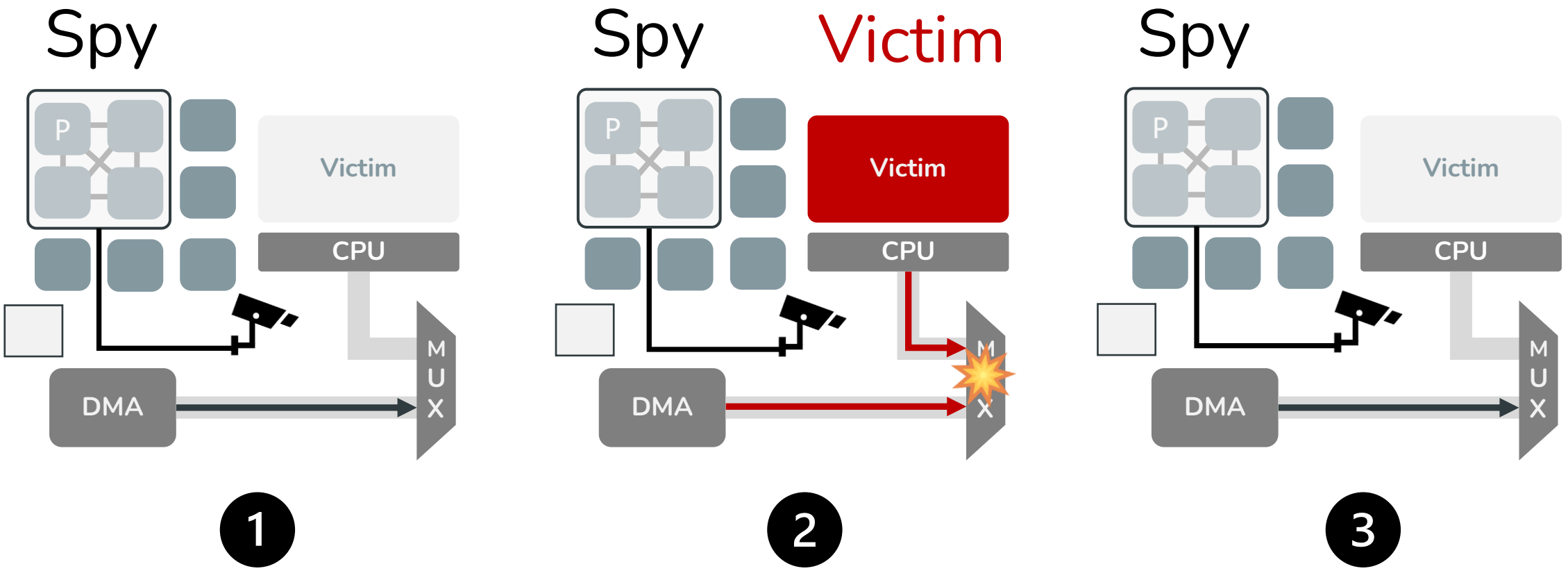
Group Some Peripherals!!

BUSted Solution



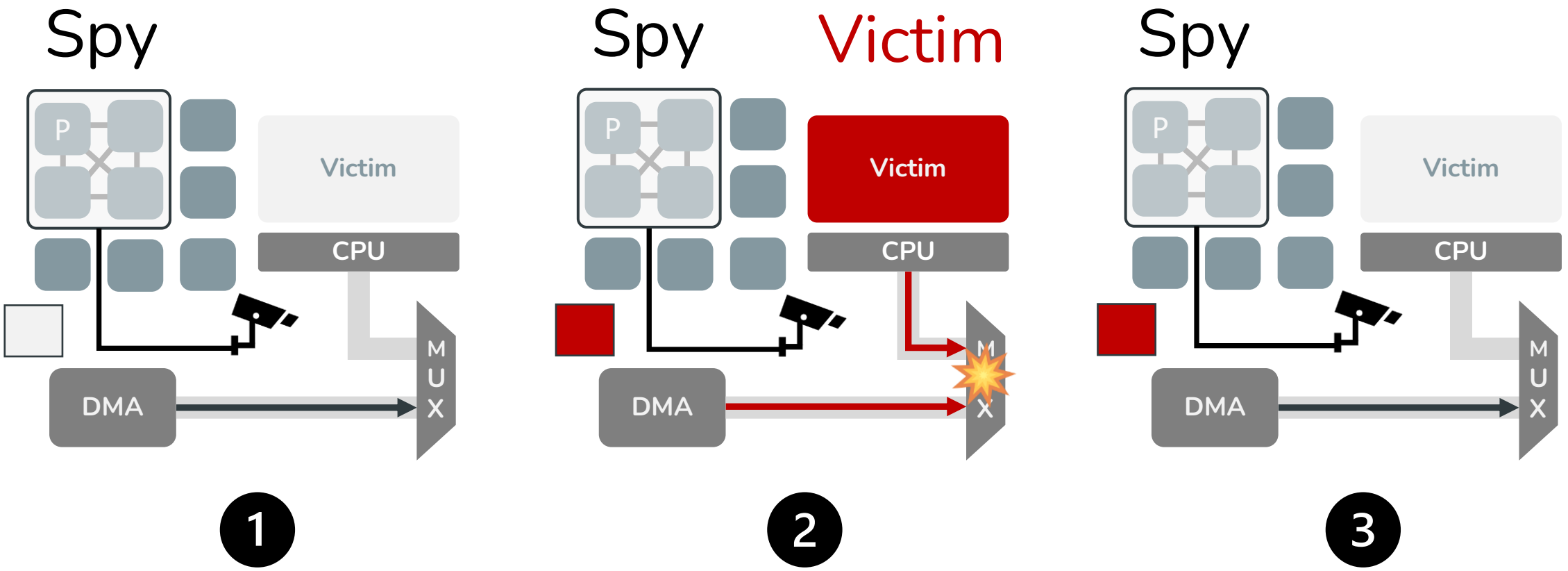
Interconnect the Peripherals!!

BUSted Solution



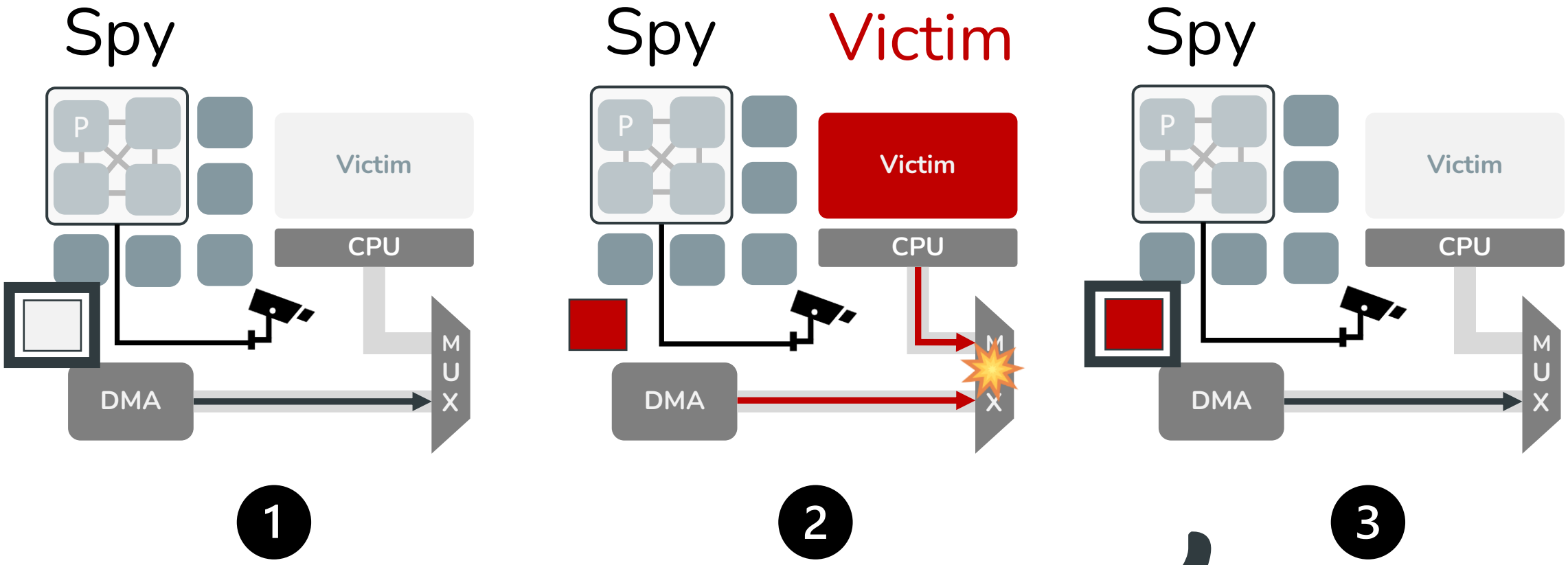
Voilà!!! Hardware Gadgets!

BUSted Solution



Snapshot of the Bus Contention

BUSted Solution



Different States

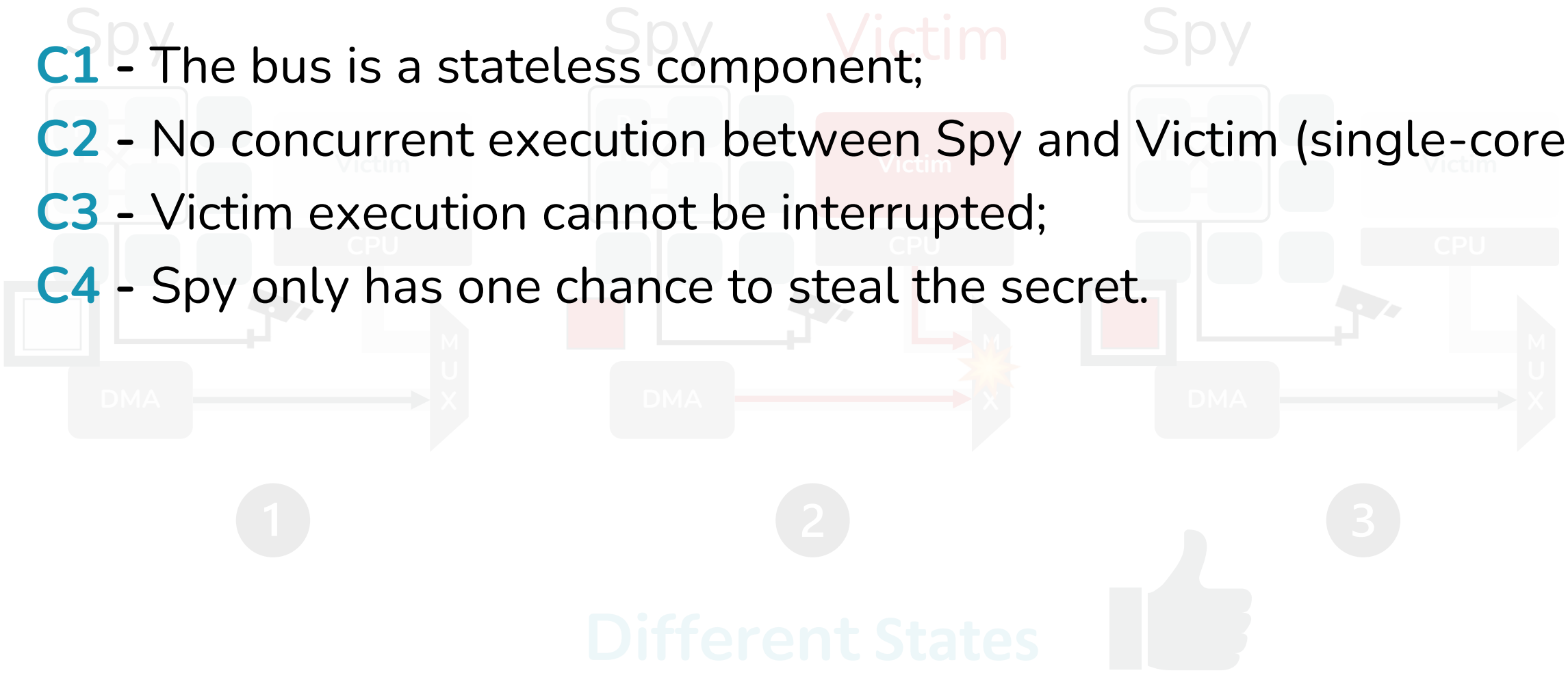
BUSted Solution

C1 - The bus is a stateless component;

C2 - No concurrent execution between Spy and Victim (single-core);

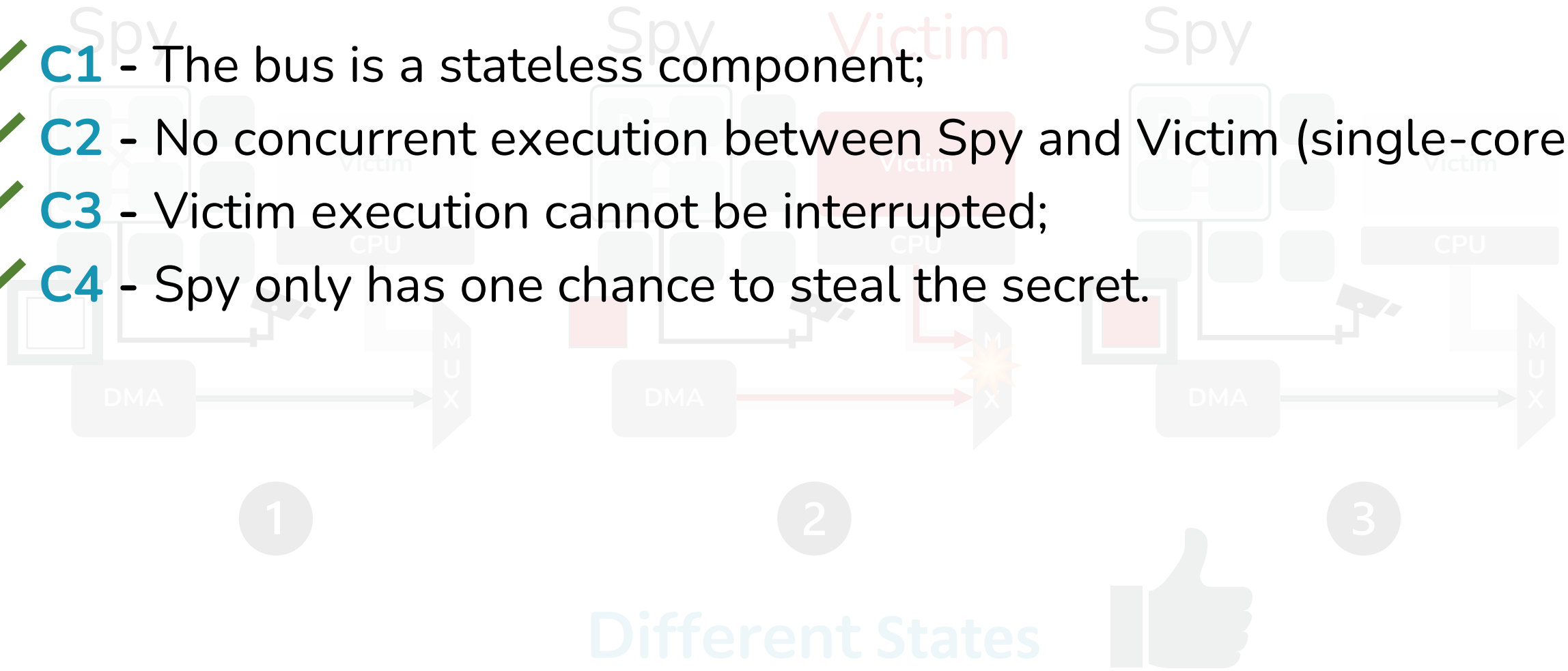
C3 - Victim execution cannot be interrupted;

C4 - Spy only has one chance to steal the secret.

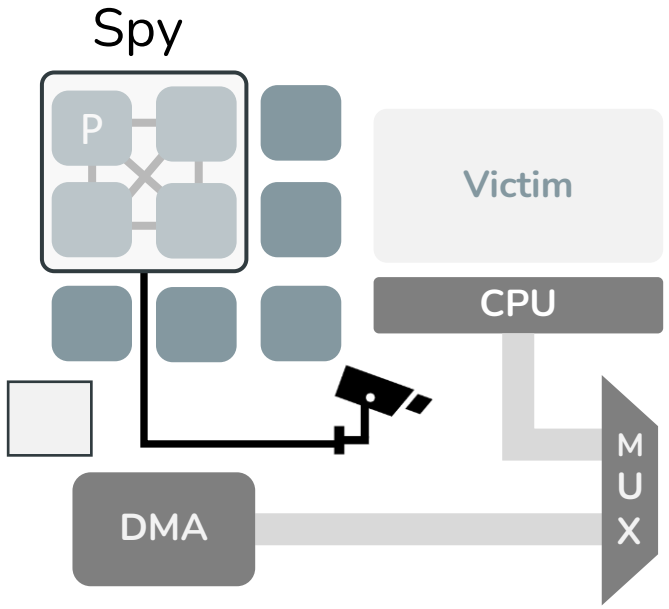


BUSted Solution

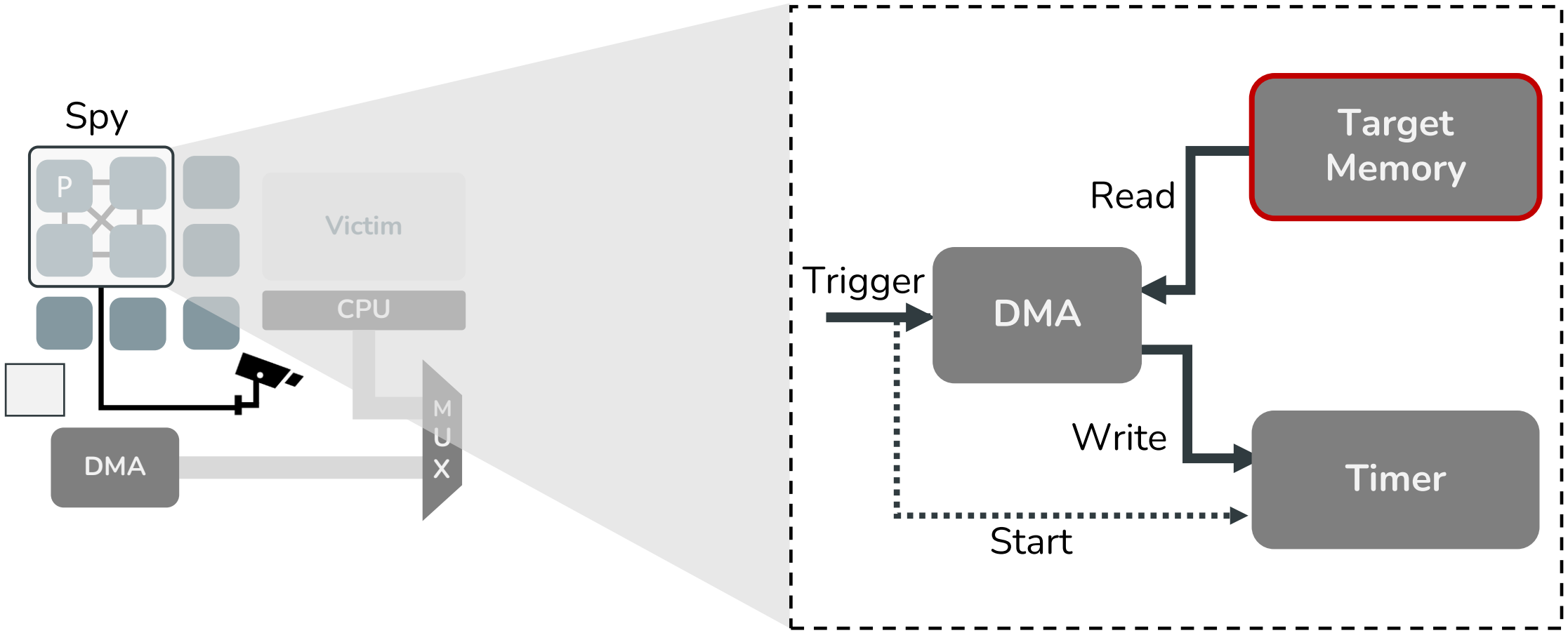
- ✓ **C1** - The bus is a stateless component;
- ✓ **C2** - No concurrent execution between Spy and Victim (single-core);
- ✓ **C3** - Victim execution cannot be interrupted;
- ✓ **C4** - Spy only has one chance to steal the secret.



Hardware Gadgets

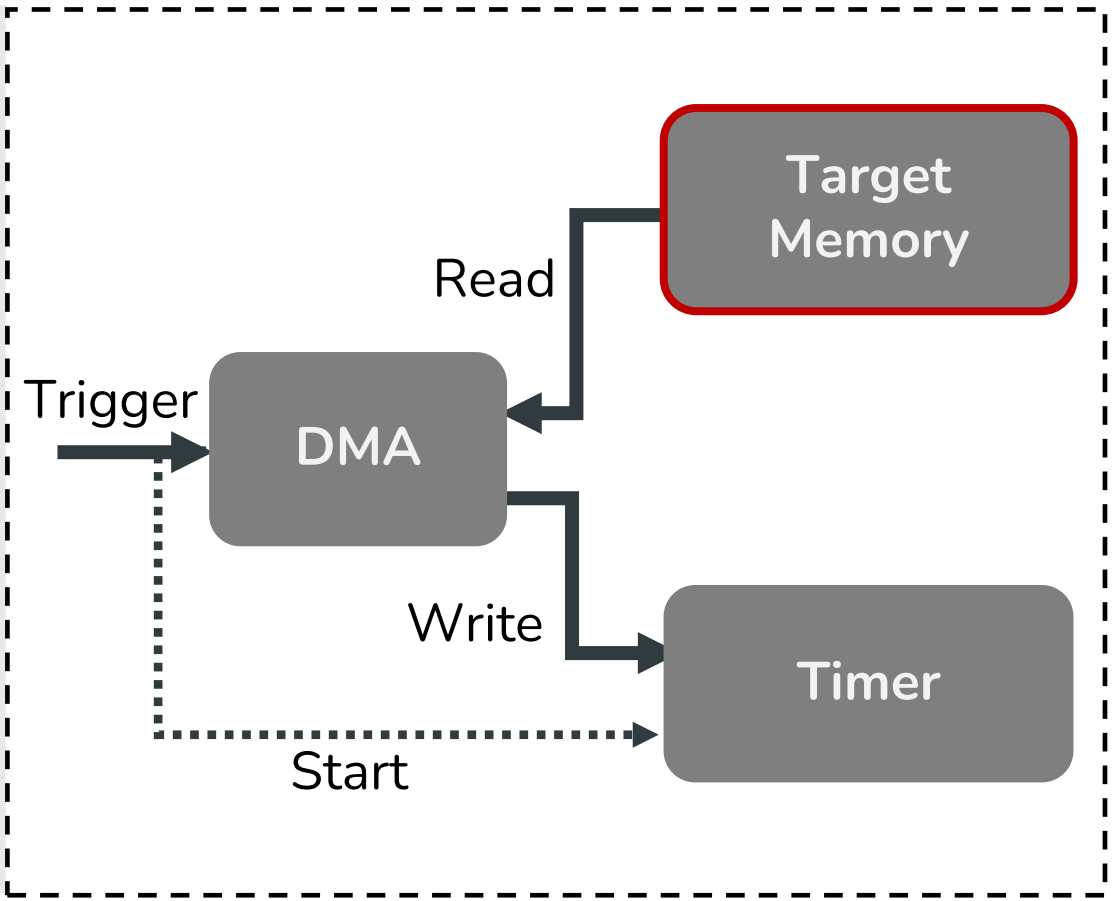


From Software to Hardware



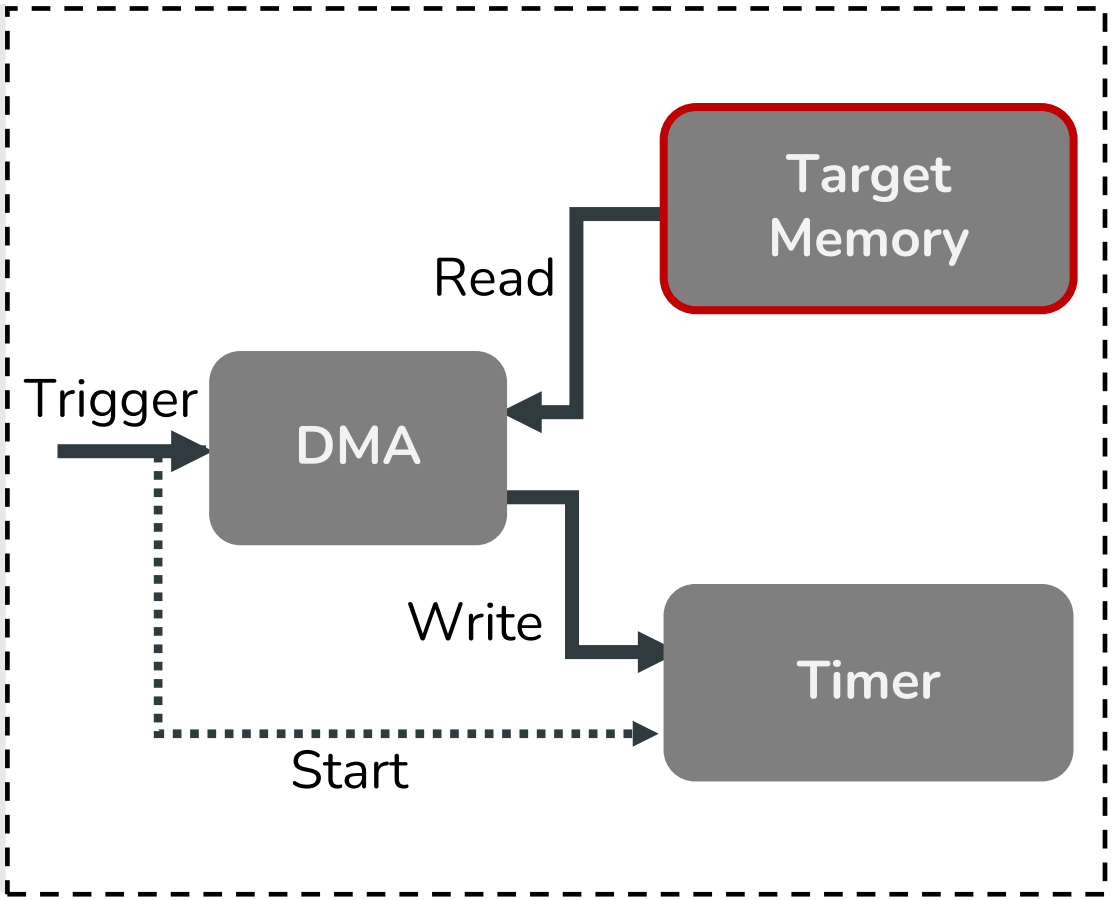
From Software to Hardware

```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;

    start_time = get_time();
    *dst = *src;
    end_time = get_time();

    access_time = end_time - start_time;

    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```



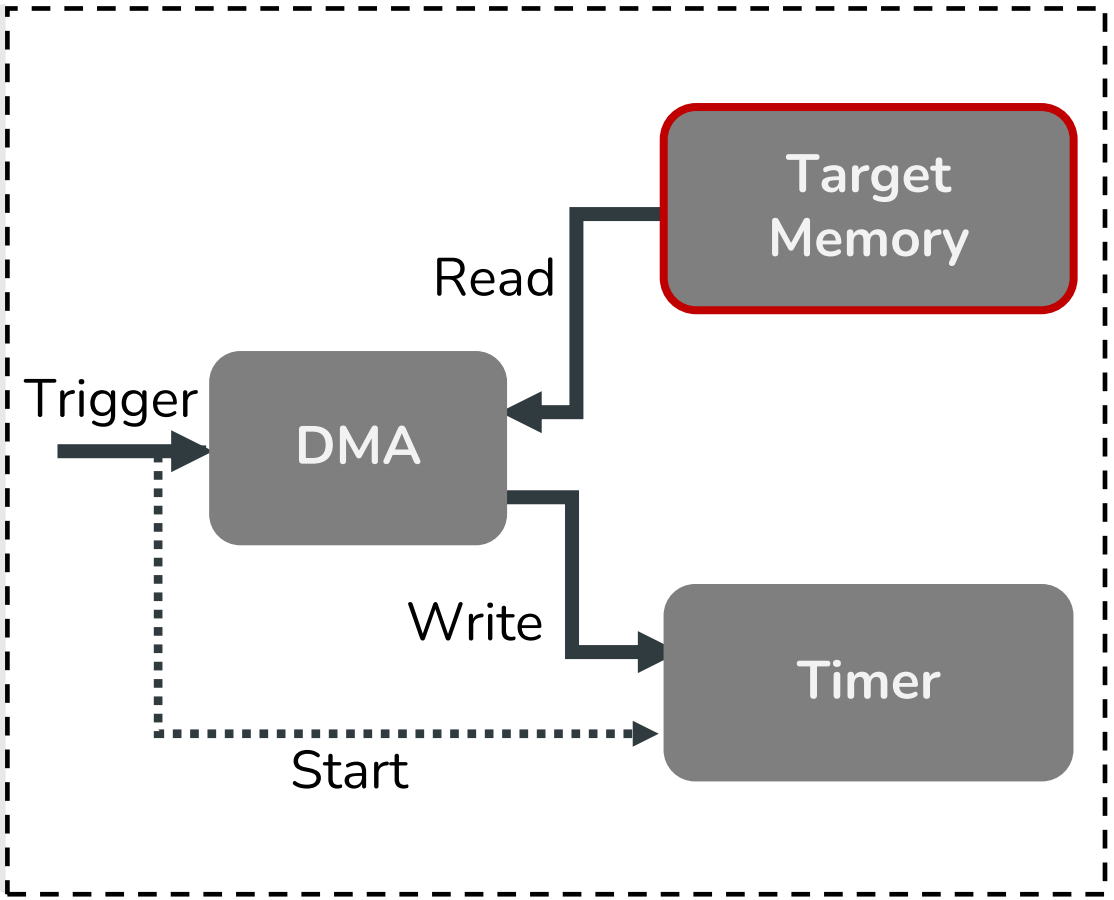
From Software to Hardware

```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;

    start_time = get_time();
    *dst = *src;
    end_time = get_time();

    access_time = end_time - start_time;

    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```



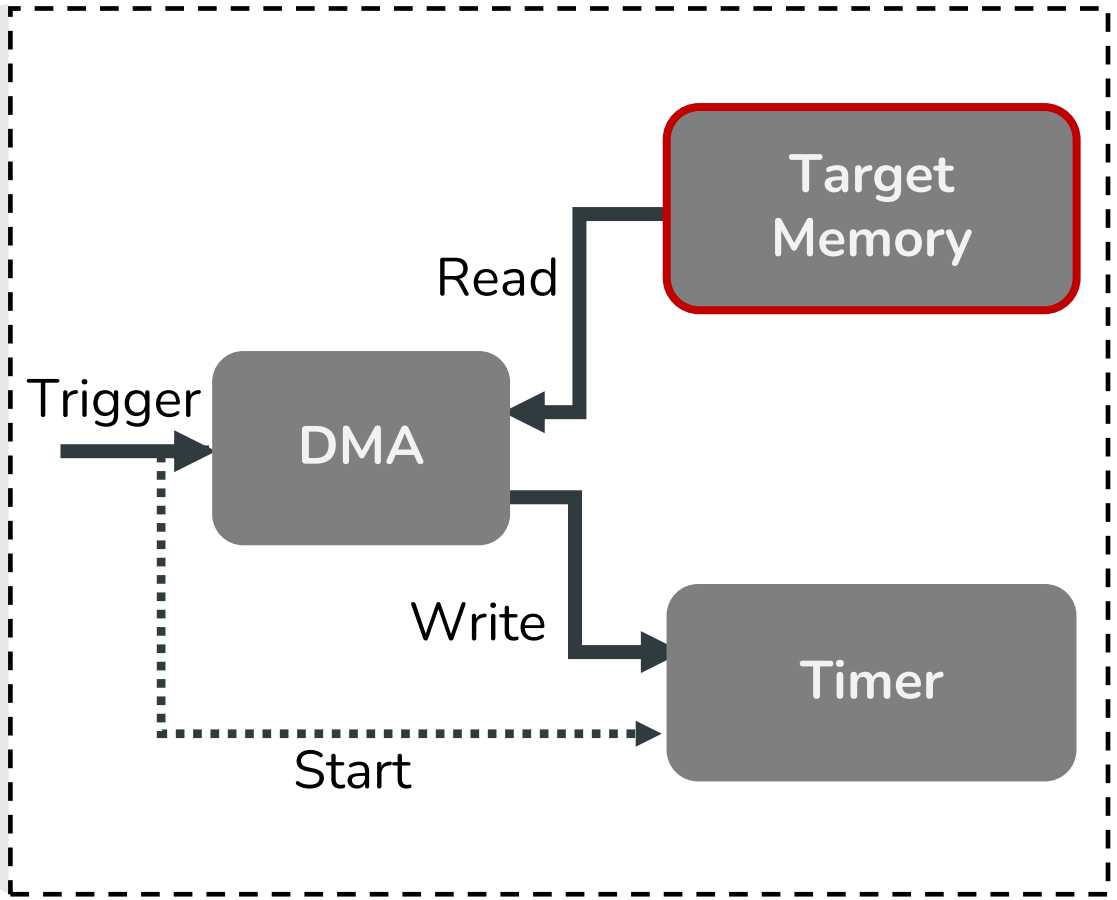
From Software to Hardware

```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;

    start_time = get_time();
    *dst = *src;
    end_time = get_time();

    access_time = end_time - start_time;

    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```



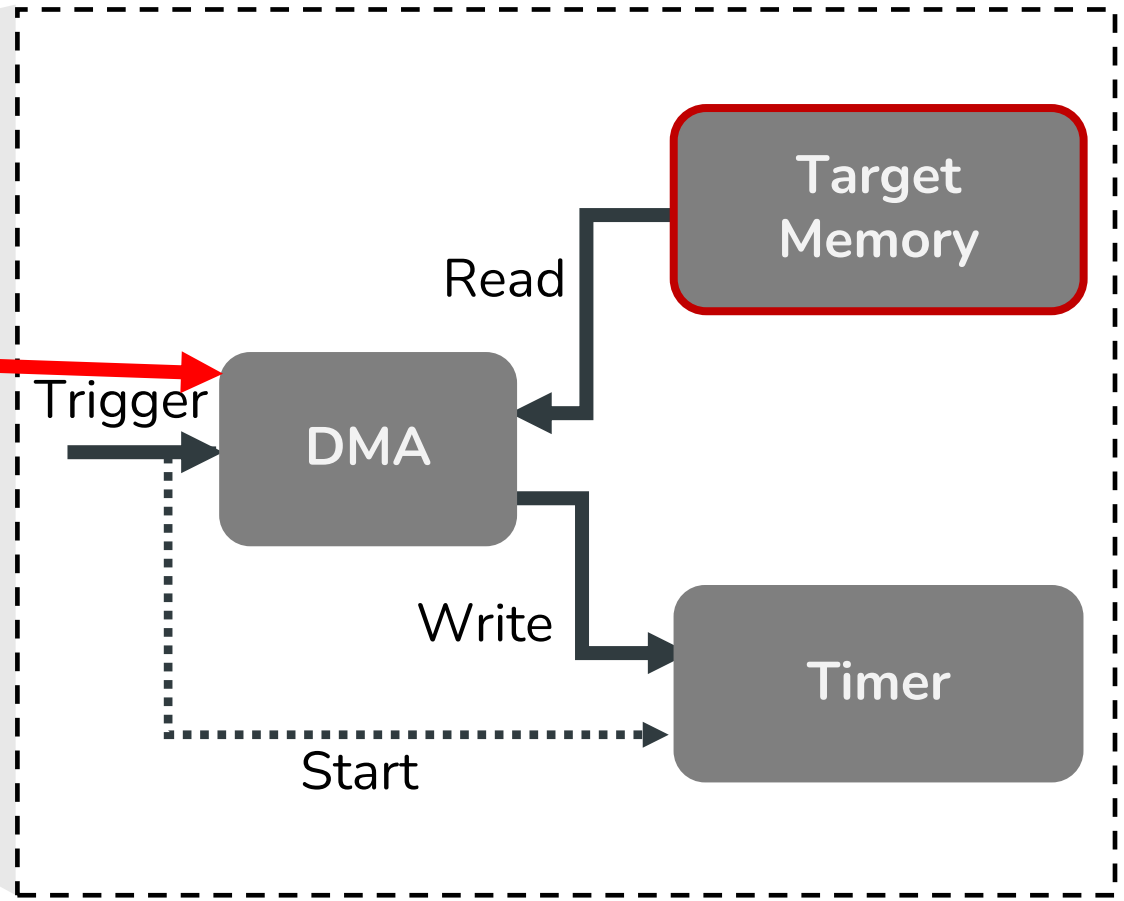
From Software to Hardware

```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;

    start_time = get_time();
    *dst = *src;
    end_time = get_time();

    access_time = end_time - start_time;

    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```



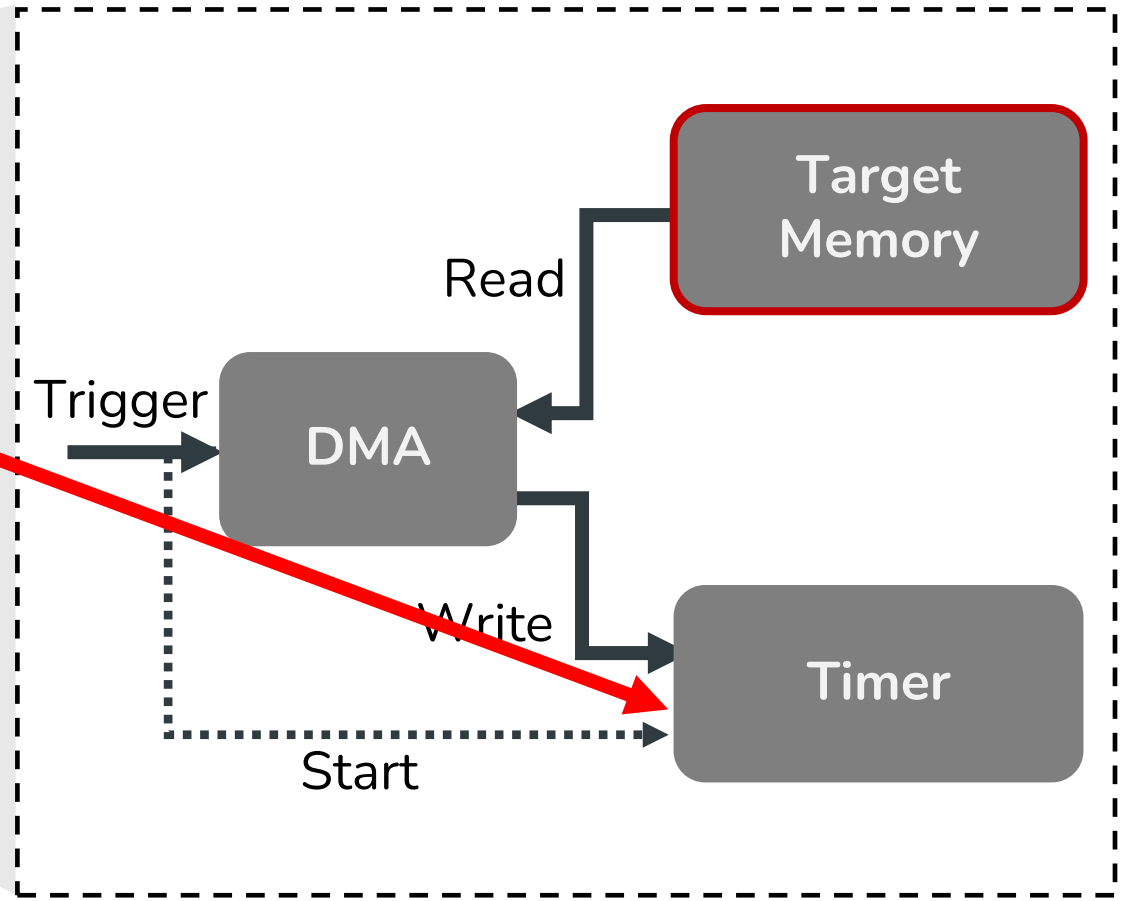
From Software to Hardware

```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;

    start_time = get_time();
    *dst = *src;
    end_time = get_time();

    access_time = end_time - start_time;

    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```



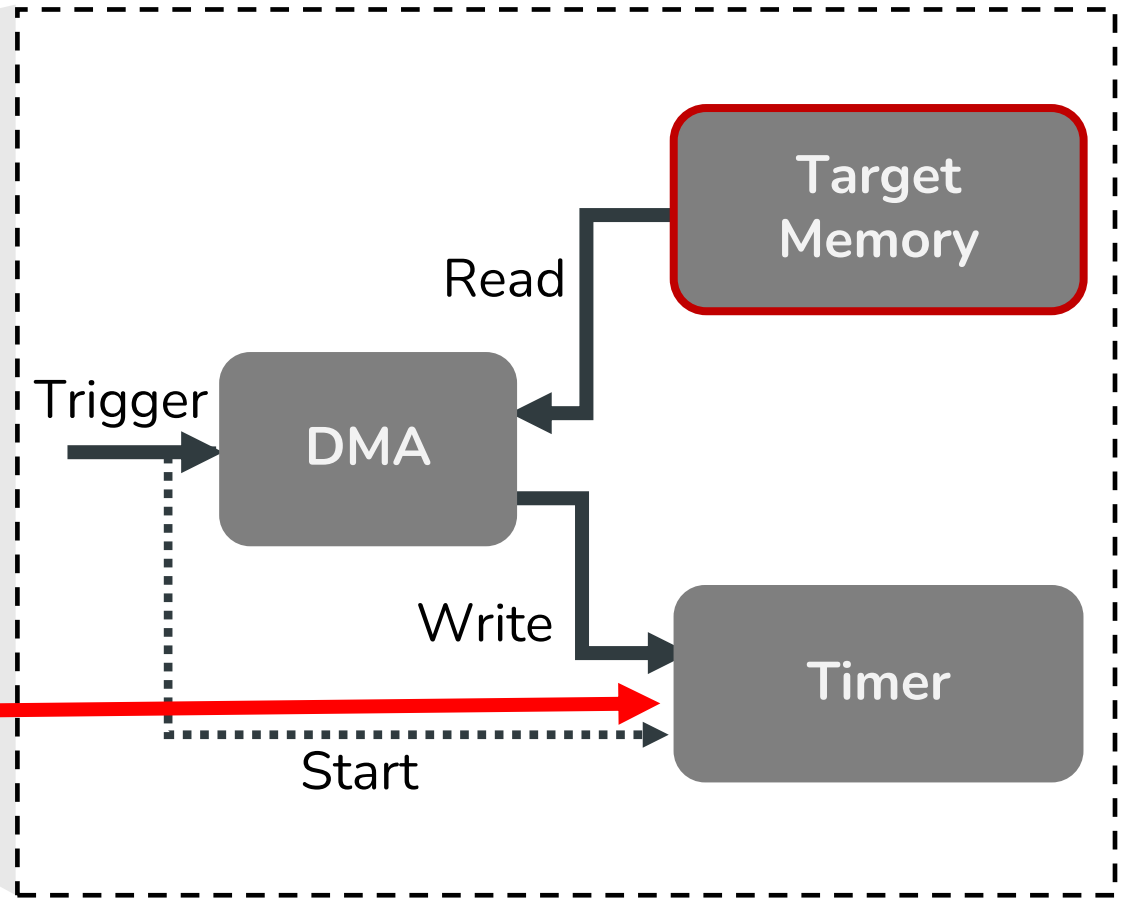
From Software to Hardware

```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;

    start_time = get_time();
    *dst = *src;
    end_time = get_time();

    access_time = end_time - start_time;

    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```



From Software to Hardware

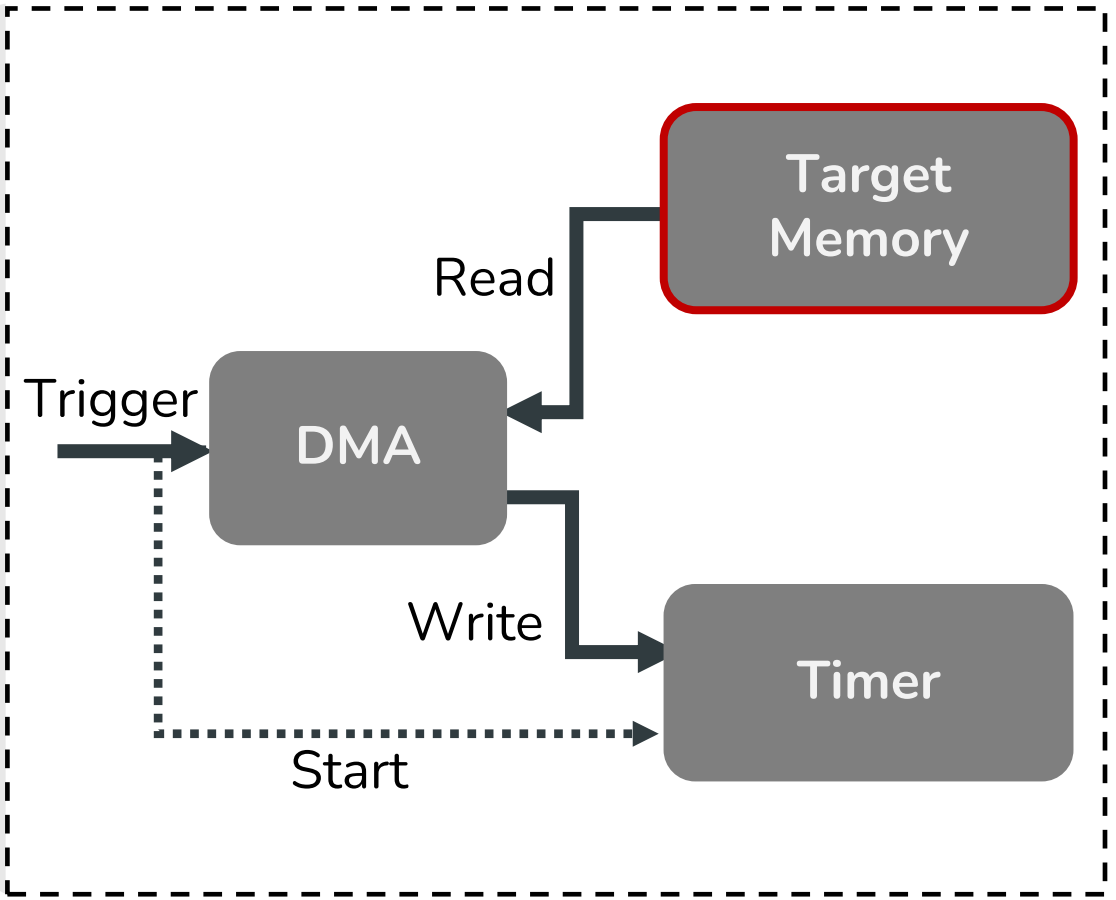
```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;

    start_time = get_time();
    *dst = *src;
    end_time = get_time();

    access_time = end_time - start_time;

    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```



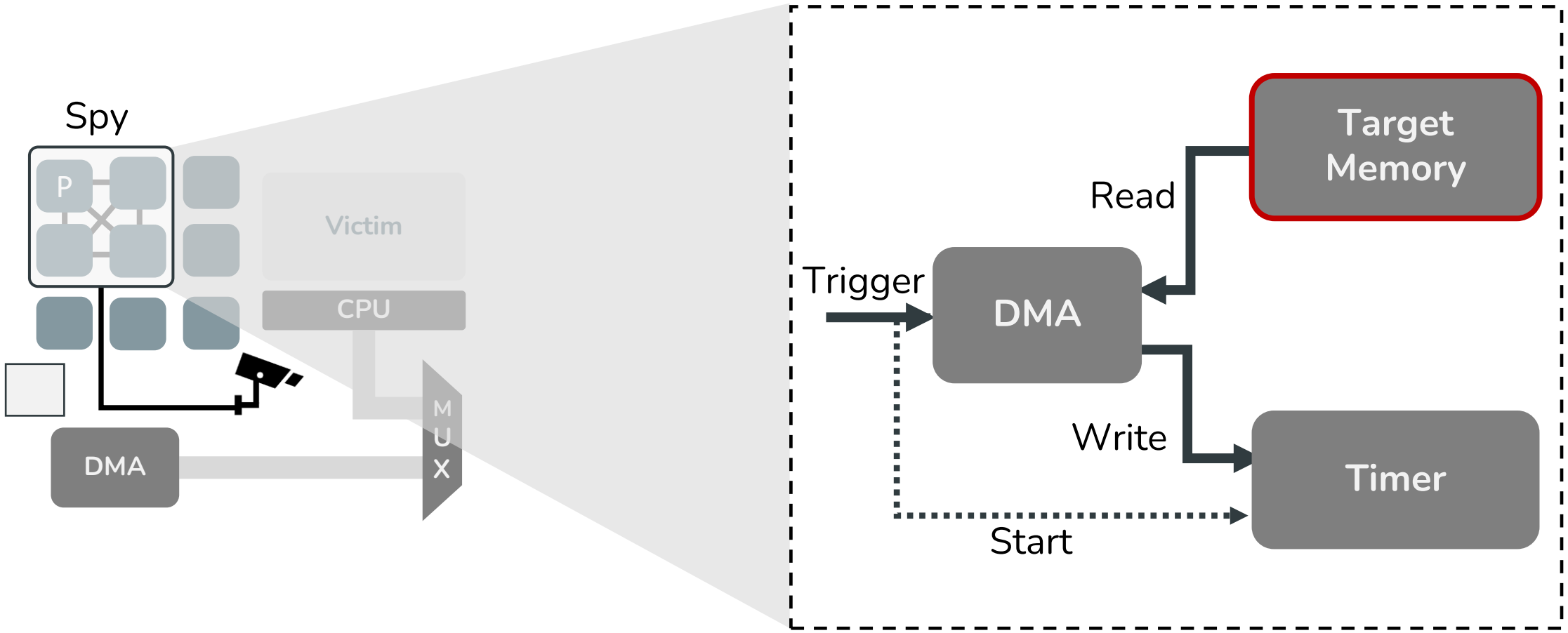
From Software to Hardware

```
#define CONTENTION_THRESHOLD 20
uint8_t detect_contention(){
    uint32_t *src = addr1, *dst = addr2;
    uint32_t start_time, end_time,
             access_time, contention;
    start_time = get_time();
    *dst = *src;
    end_time = get_time();
    access_time = end_time - start_time;
    if(access_time > CONTENTION_THRESHOLD)
        contention = 1;
    else
        contention = 0;
    return contention;
}
```

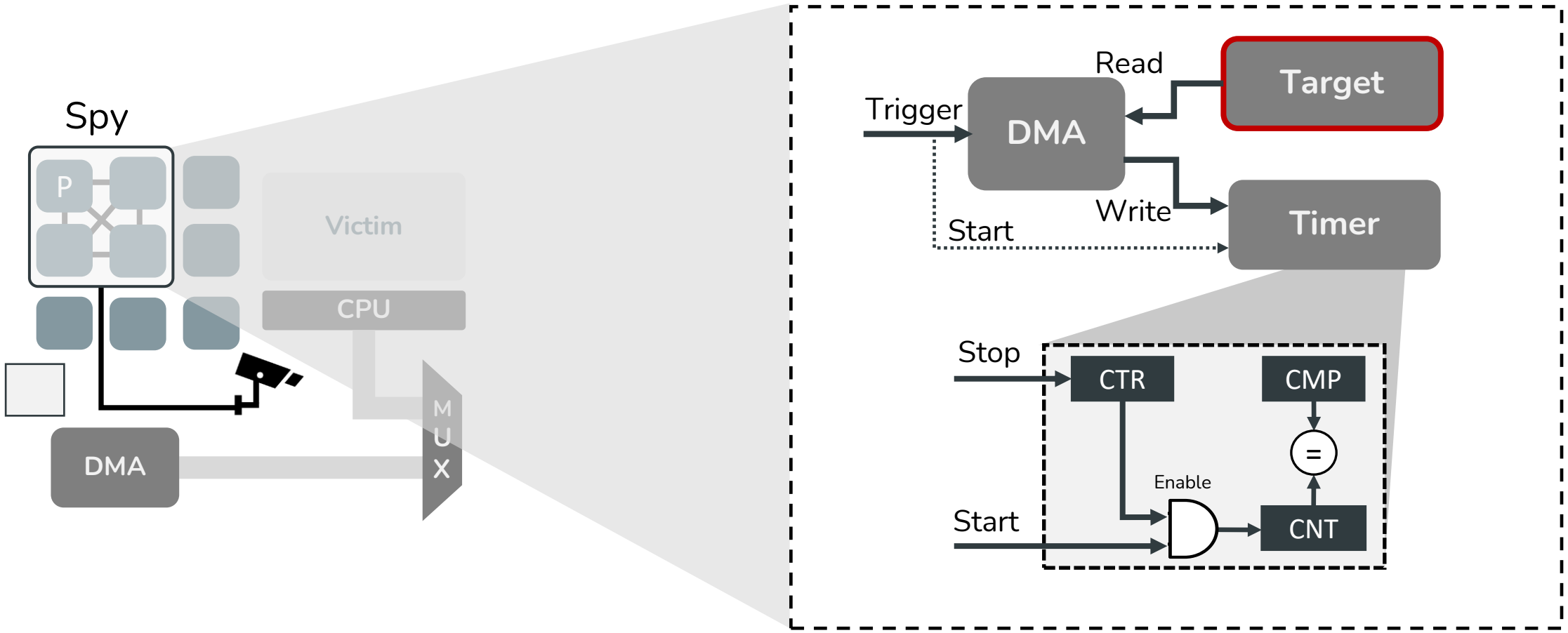
Detect Contention Offloaded to a **Hardware Gadget**



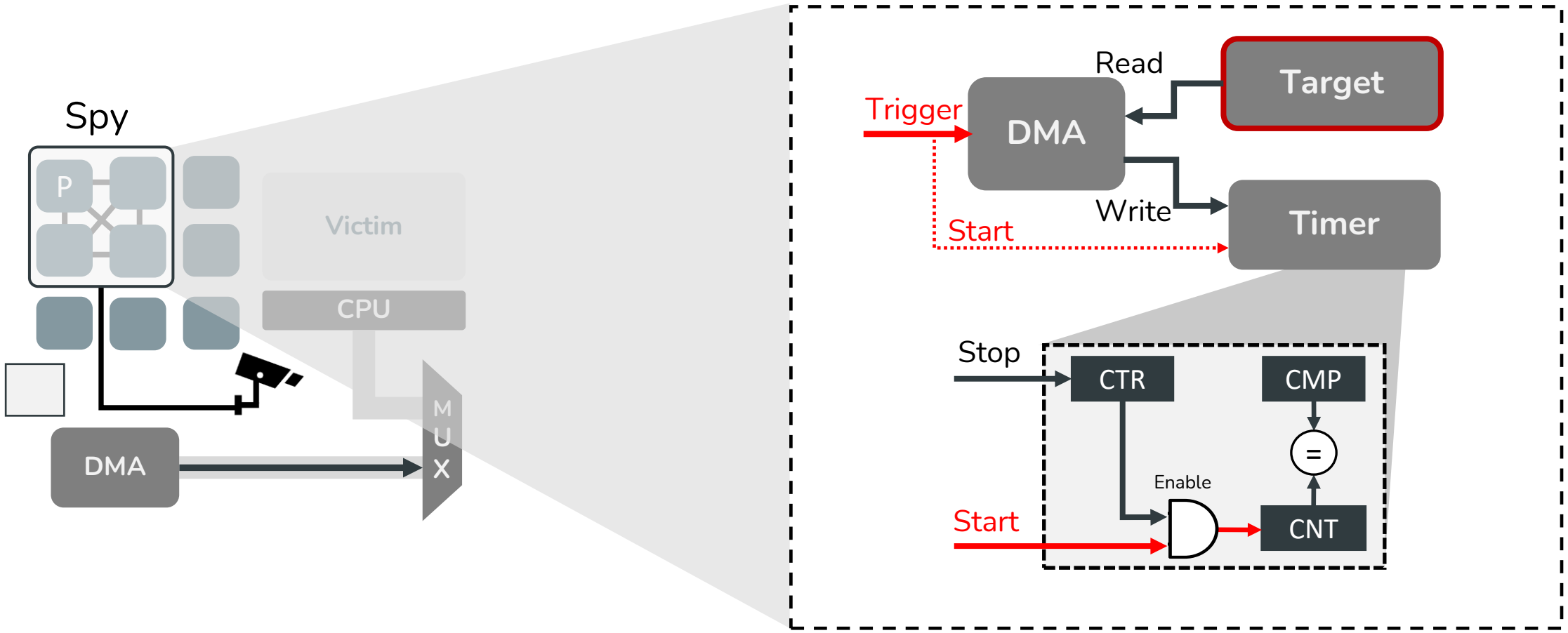
Hardware Gadgets



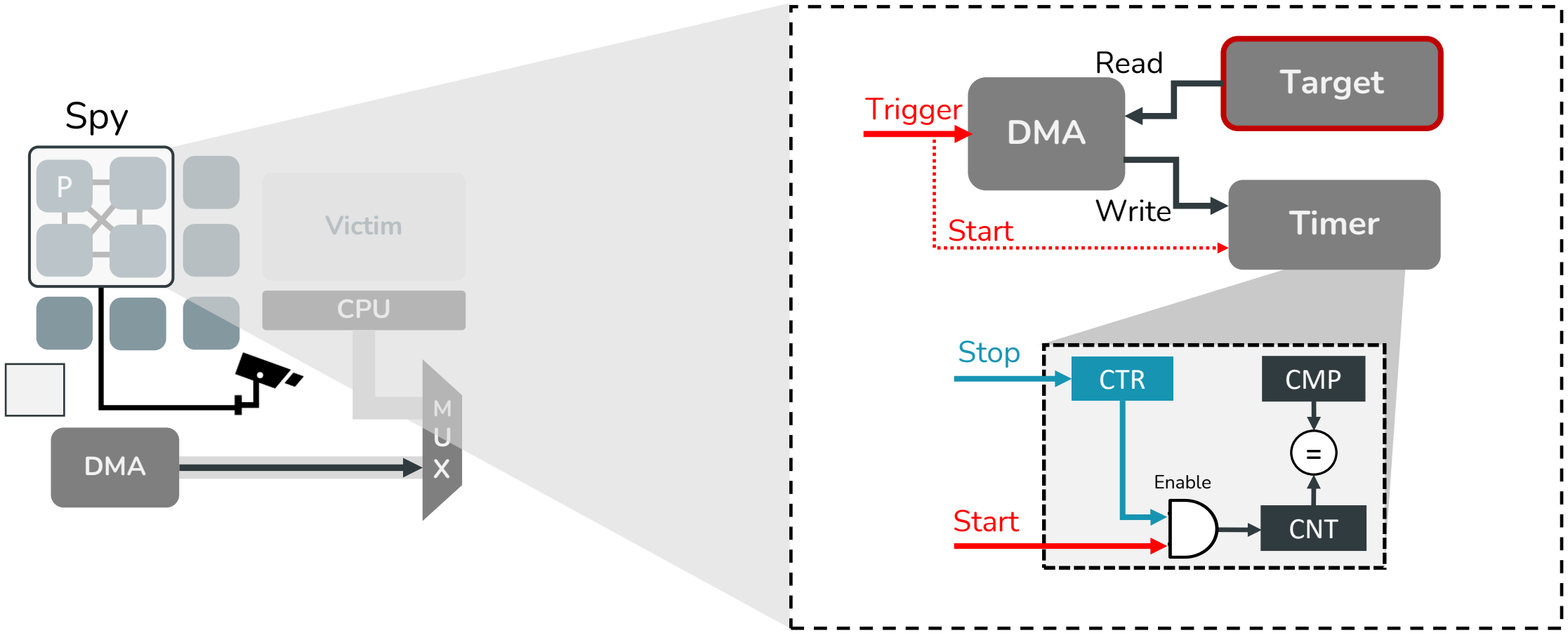
Hardware Gadgets



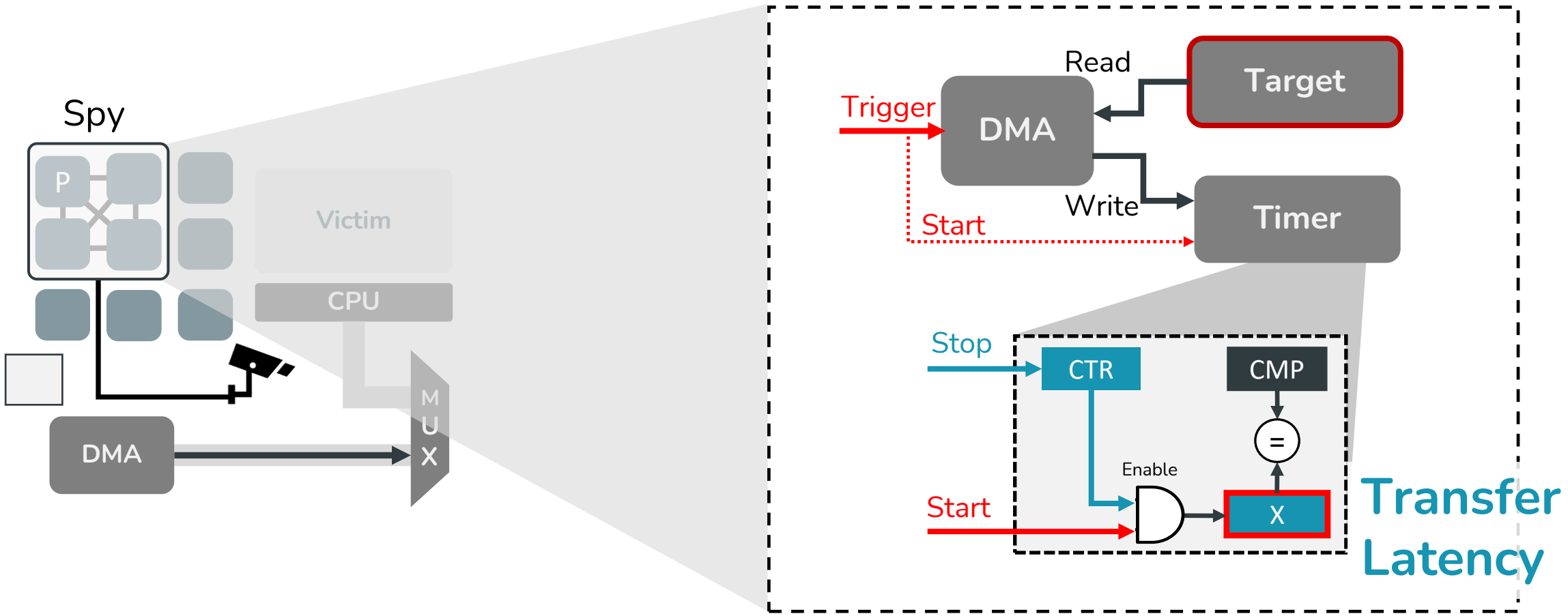
Hardware Gadgets



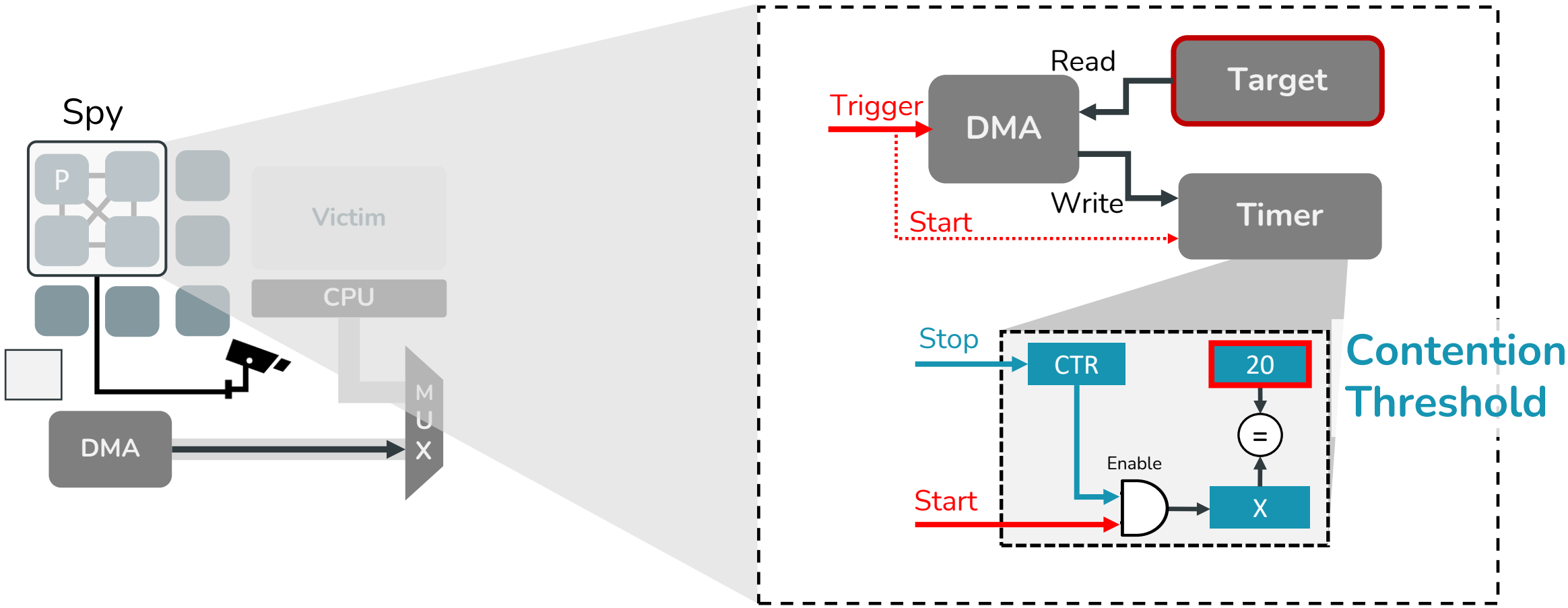
Hardware Gadgets



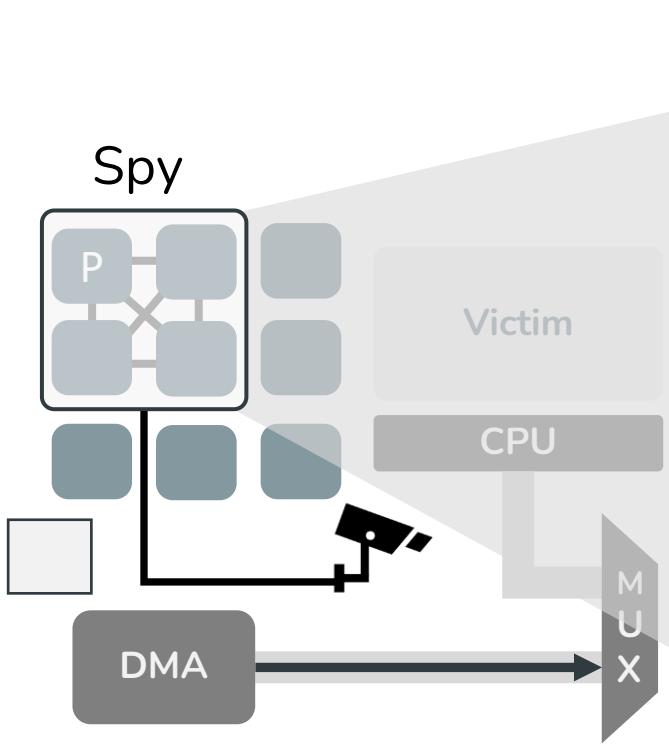
Hardware Gadgets



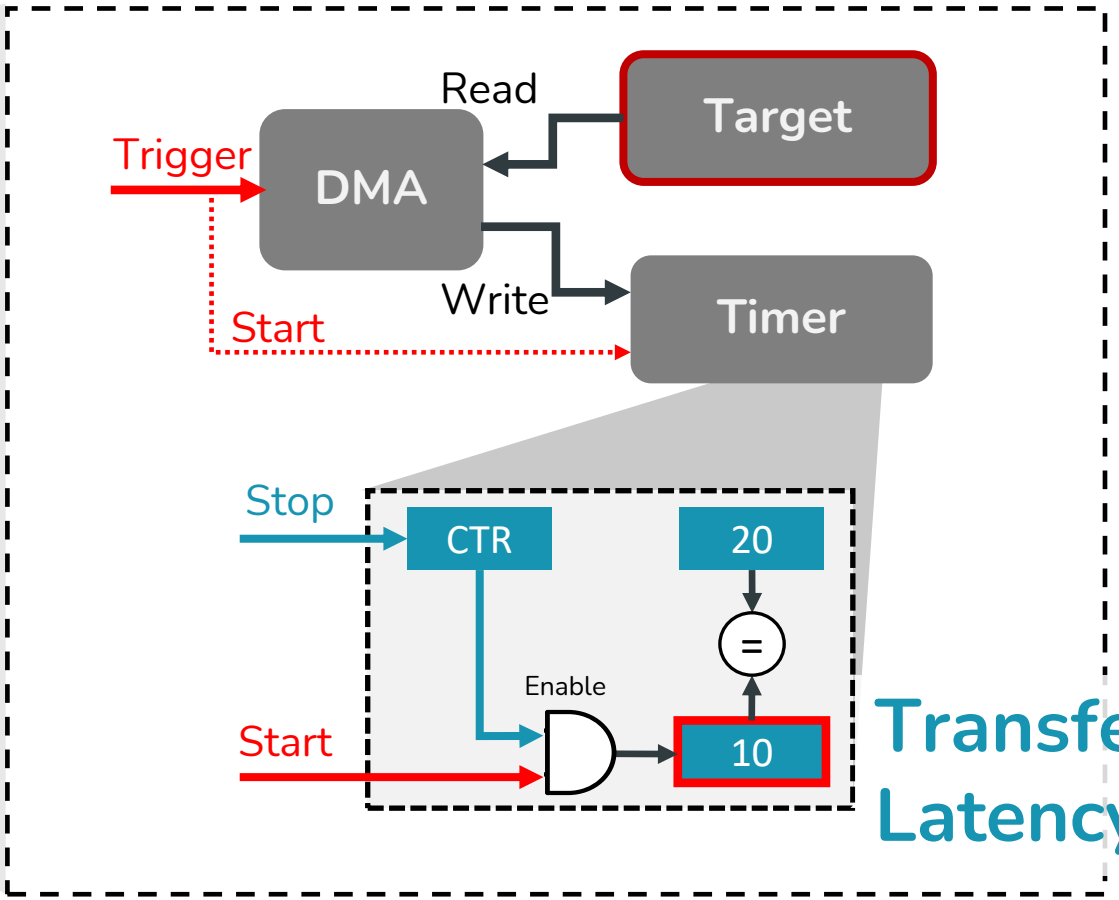
Hardware Gadgets



Hardware Gadgets

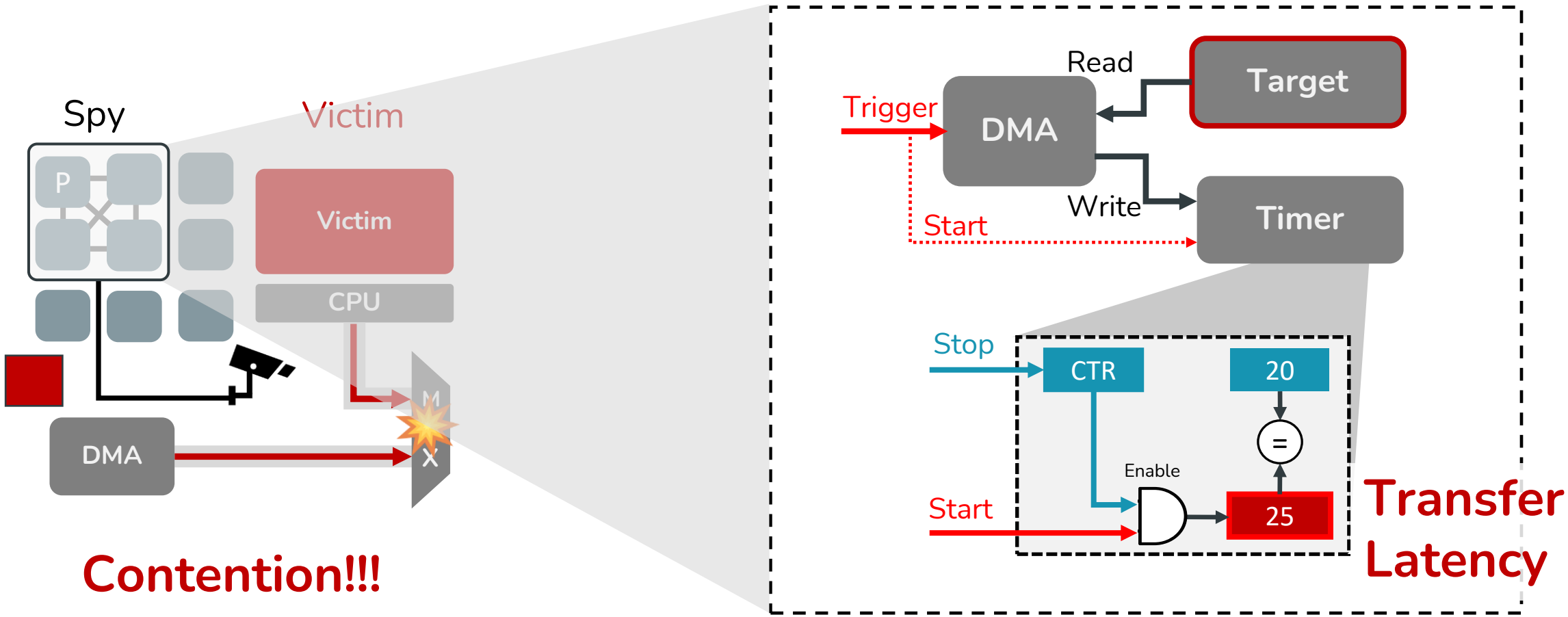


Transfer Without Contention

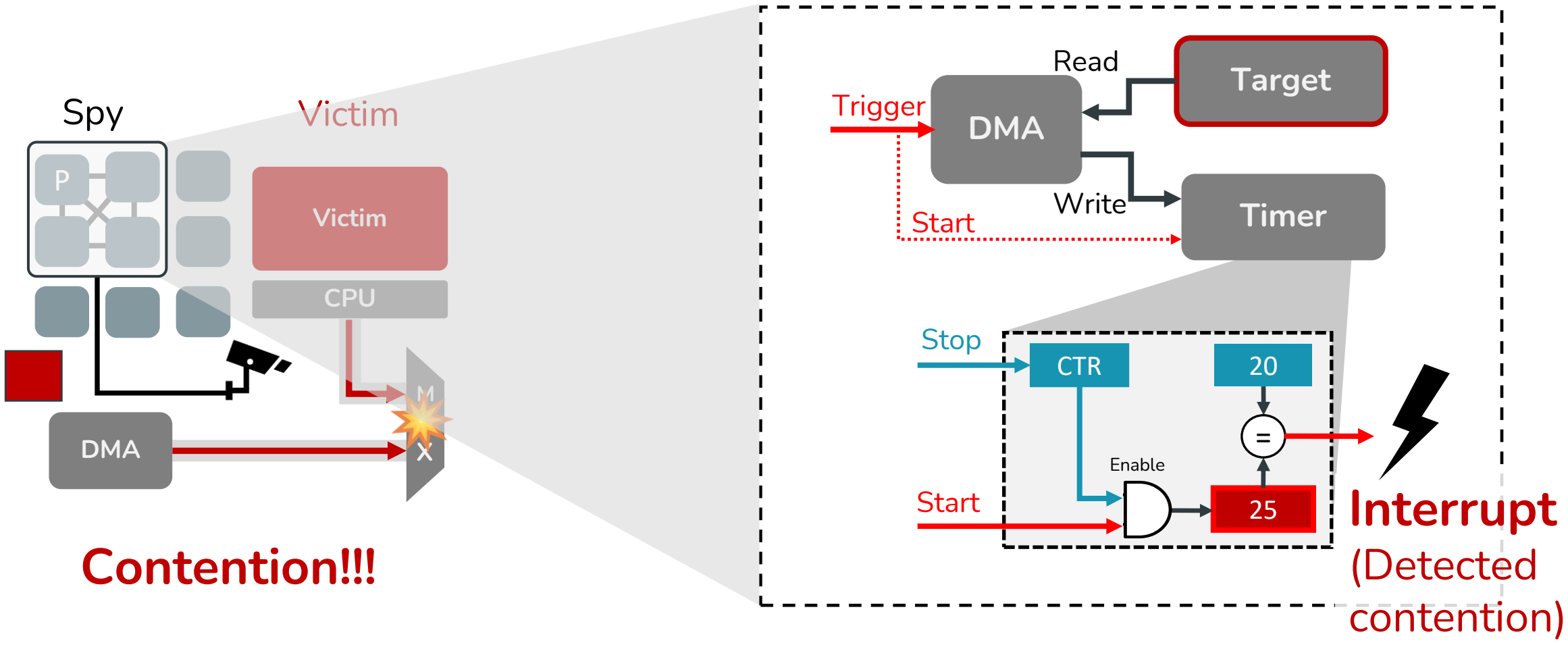


Transfer Latency

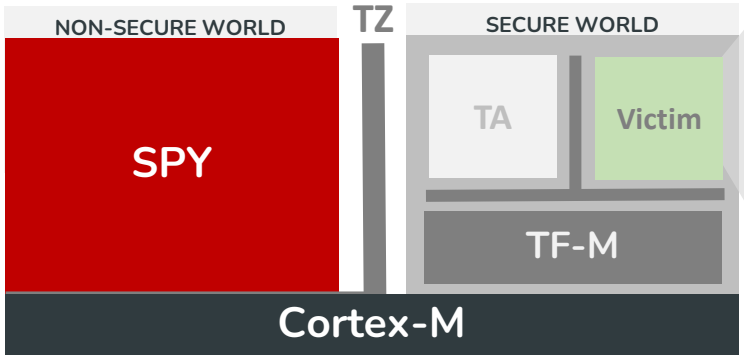
Hardware Gadgets



Hardware Gadgets



BUSted Attack

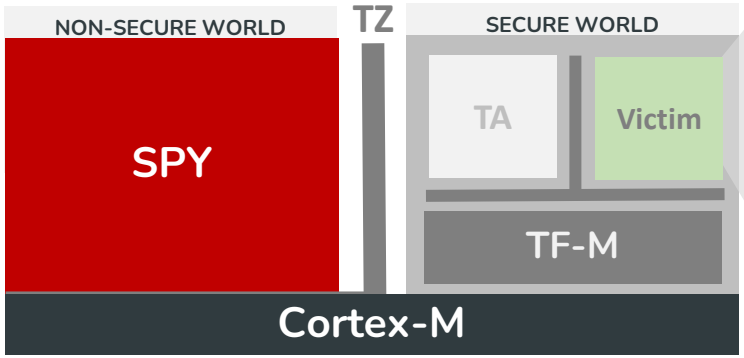



```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

Code based in Sancus and Texas Reference Implementation of a Keypad [1,2]

[1] https://github.com/sancus-tee/vulcan/blob/master/demo/ecu-tcs/sm_tcs_kypd.c
[2] Implementing An Ultra-Low-Power Keypad Interface With MSP430™ MCUs

BUSted Attack



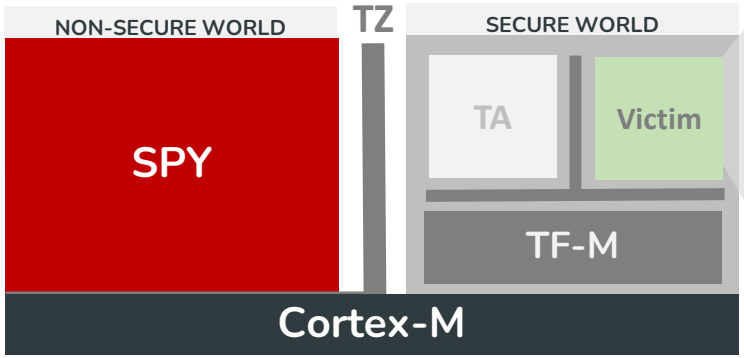

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

while loop

Code based in Sancus and Texas Reference Implementation of a Keypad [1,2]

[1] https://github.com/sancus-tee/vulcan/blob/master/demo/ecu-tcs/sm_tcs_kypd.c
[2] Implementing An Ultra-Low-Power Keypad Interface With MSP430™ MCUs

BUSted Attack

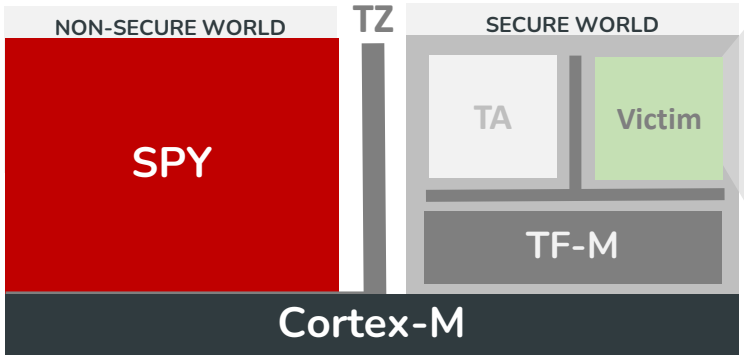



```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();                    read key
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();                                   while loop
}
```

Code based in Sancus and Texas Reference Implementation of a Keypad [1,2]

[1] https://github.com/sancus-tee/vulcan/blob/master/demo/ecu-tcs/sm_tcs_kypd.c
[2] Implementing An Ultra-Low-Power Keypad Interface With MSP430™ MCUs

BUSted Attack

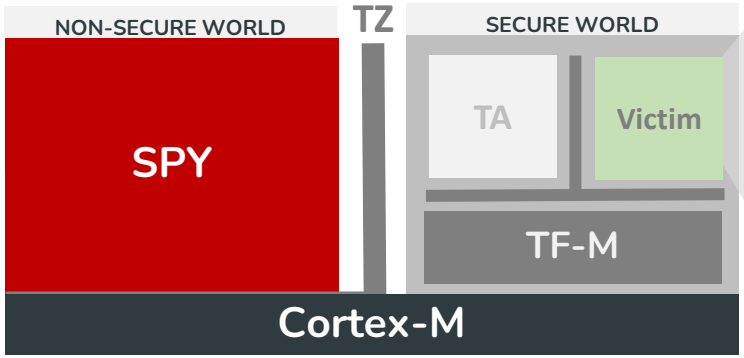



```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();                    read key
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;                                            for loop
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();                                   while loop
}
```

Code based in Sancus and Texas Reference Implementation of a Keypad [1,2]

[1] https://github.com/sancus-tee/vulcan/blob/master/demo/ecu-tcs/sm_tcs_kypd.c
[2] Implementing An Ultra-Low-Power Keypad Interface With MSP430™ MCUs

BUSted Attack

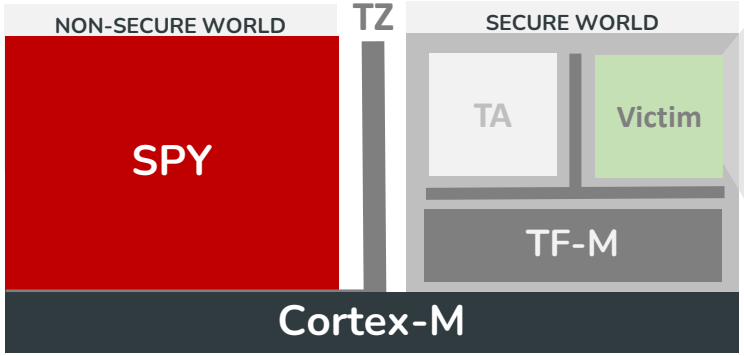



```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();                    read key
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;                                if branch
        else
          dummy_pin[dummy_pin_idx++] = key;                    else branch
        dummy_pin_idx = 0;
        mask <<= 1;                                            for loop
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();                                   while loop
}
```

Code based in Sancus and Texas Reference Implementation of a Keypad [1,2]

[1] https://github.com/sancus-tee/vulcan/blob/master/demo/ecu-tcs/sm_tcs_kypd.c
[2] Implementing An Ultra-Low-Power Keypad Interface With MSP430™ MCUs

BUSted Attack

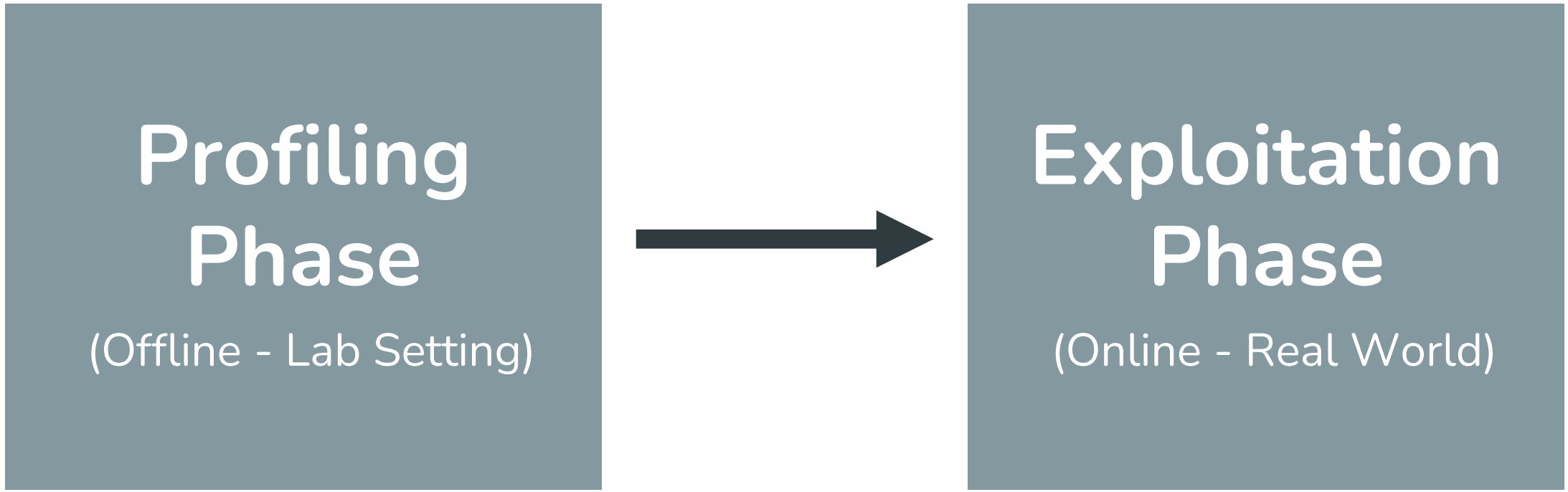



```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();                    read key
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)                                        if branch (leak)
          pin[pin_idx++] = key;
        else                                                   else branch
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;                                            for loop
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();                                   while loop
}
```

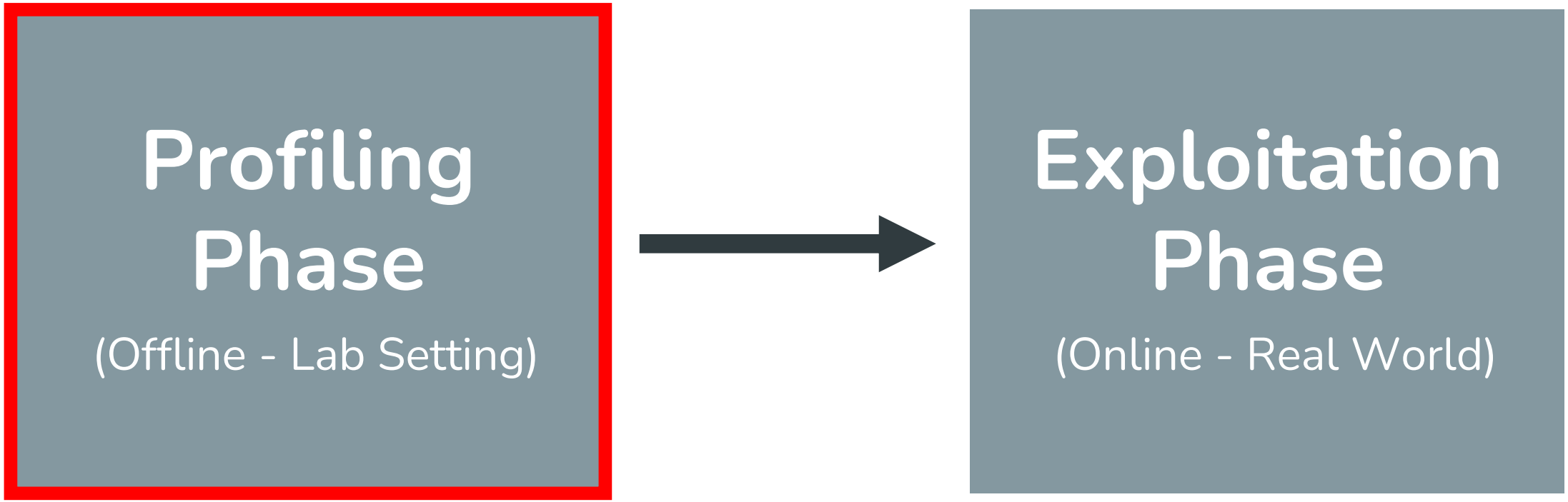
Code based in Sancus and Texas Reference Implementation of a Keypad [1,2]

[1] https://github.com/sancus-tee/vulcan/blob/master/demo/ecu-tcs/sm_tcs_kypd.c
[2] Implementing An Ultra-Low-Power Keypad Interface With MSP430™ MCUs

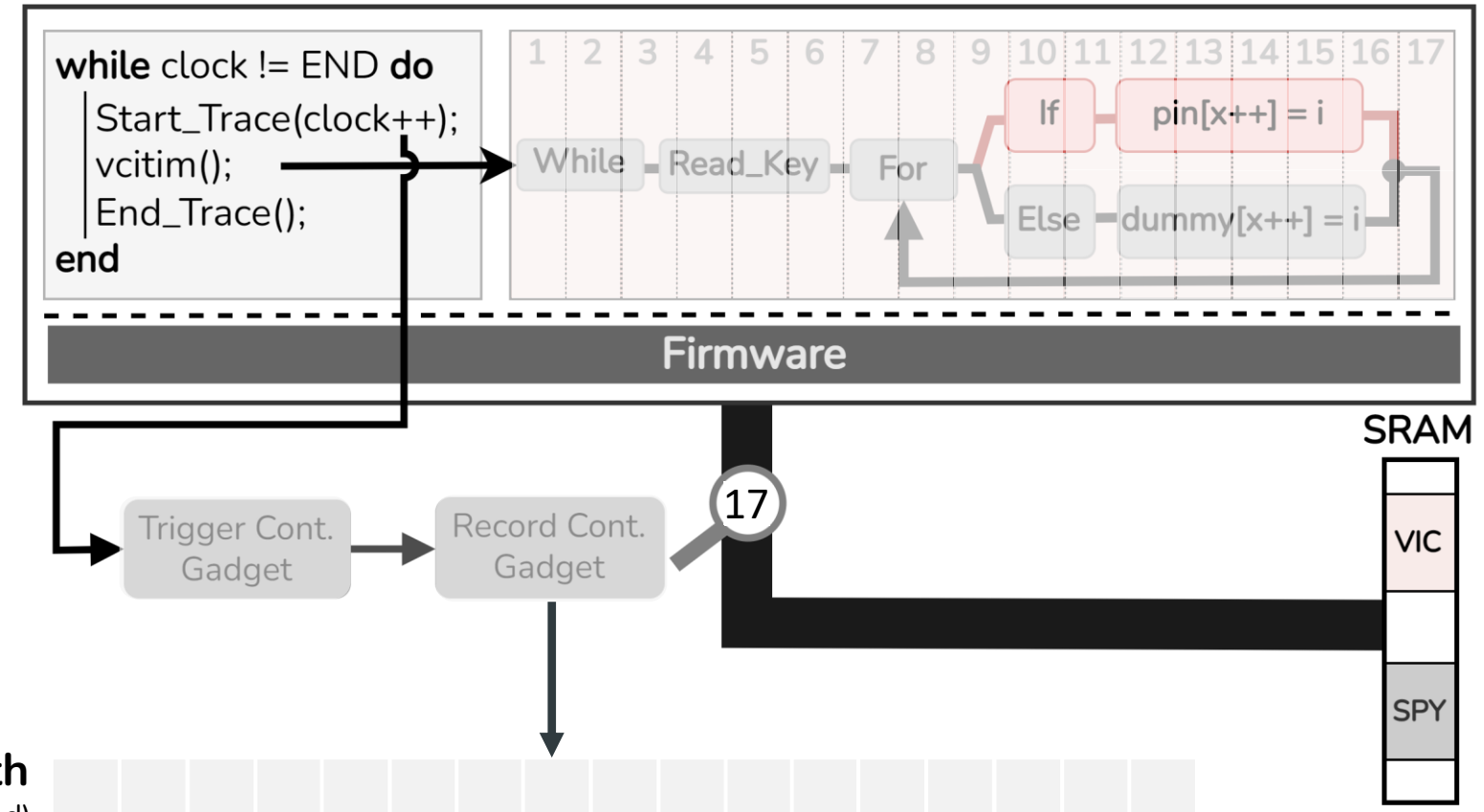
BUSted Attack Phases



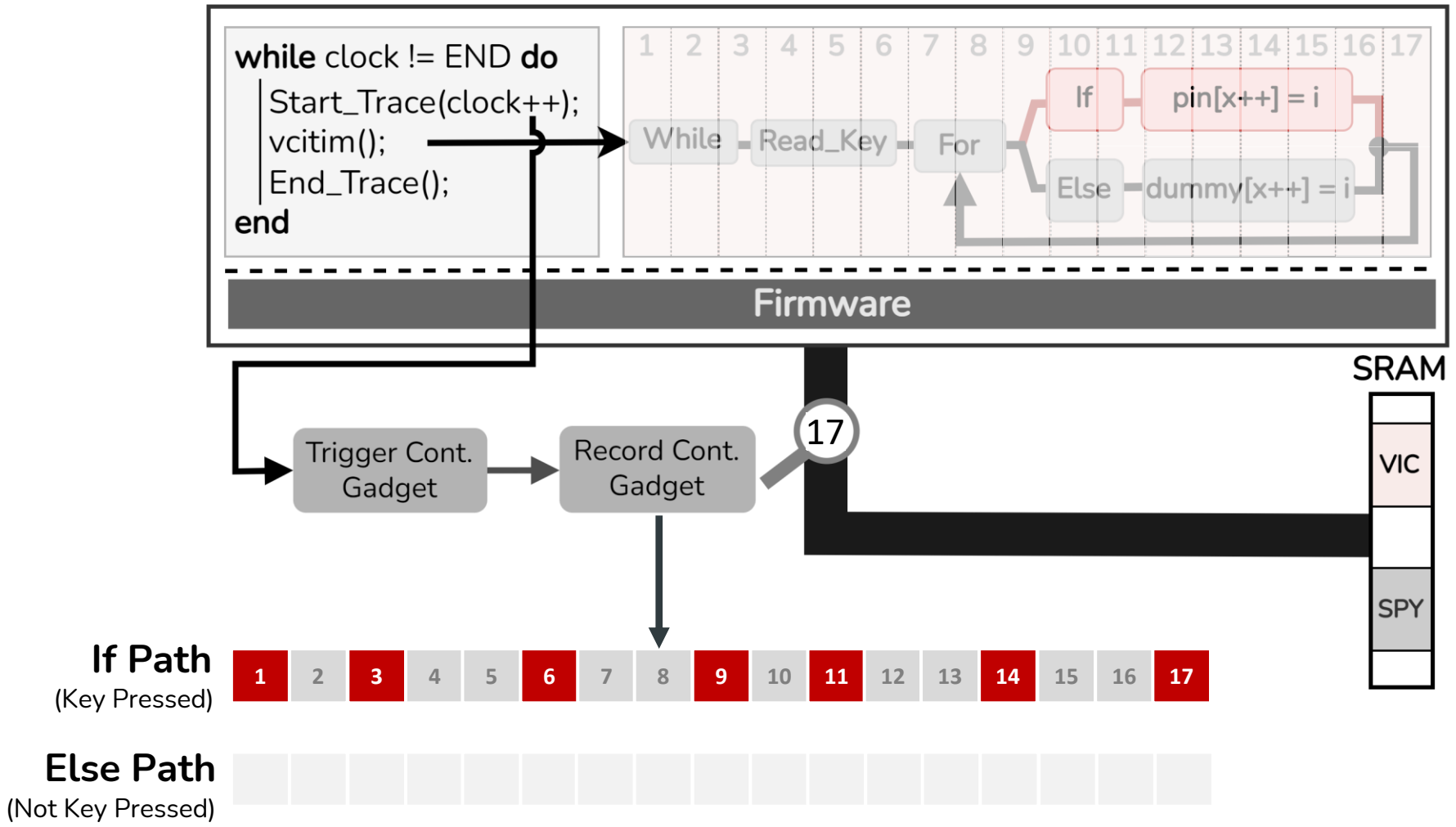
BUSted Attack Phases



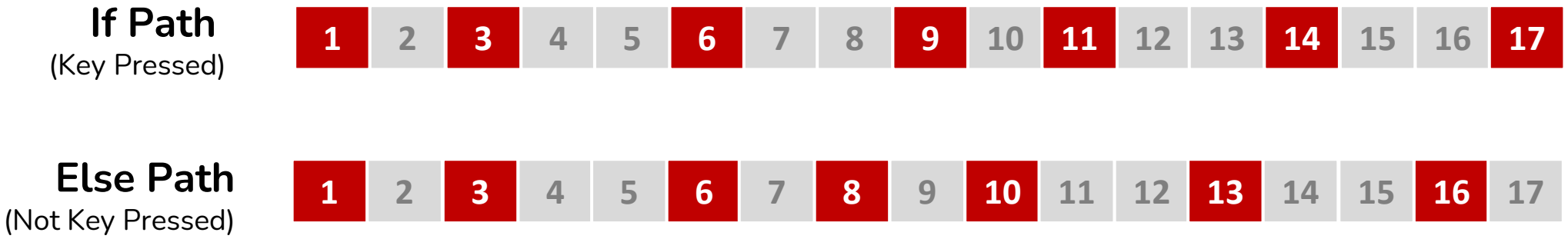
BUSted Profiling Phase



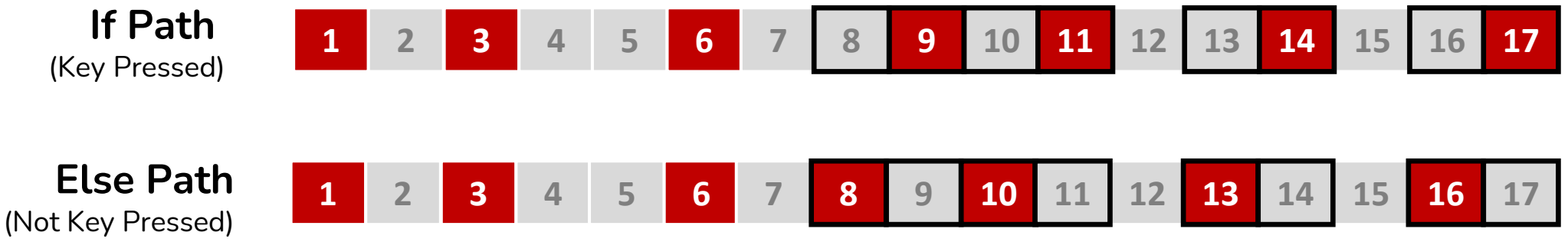
BUSted Profiling Phase



BUSted Profiling Phase

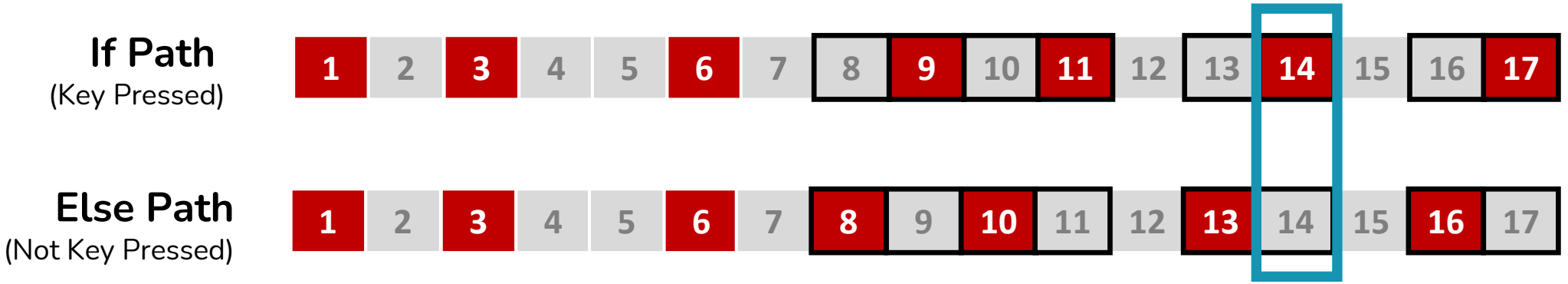


BUSted Profiling Phase



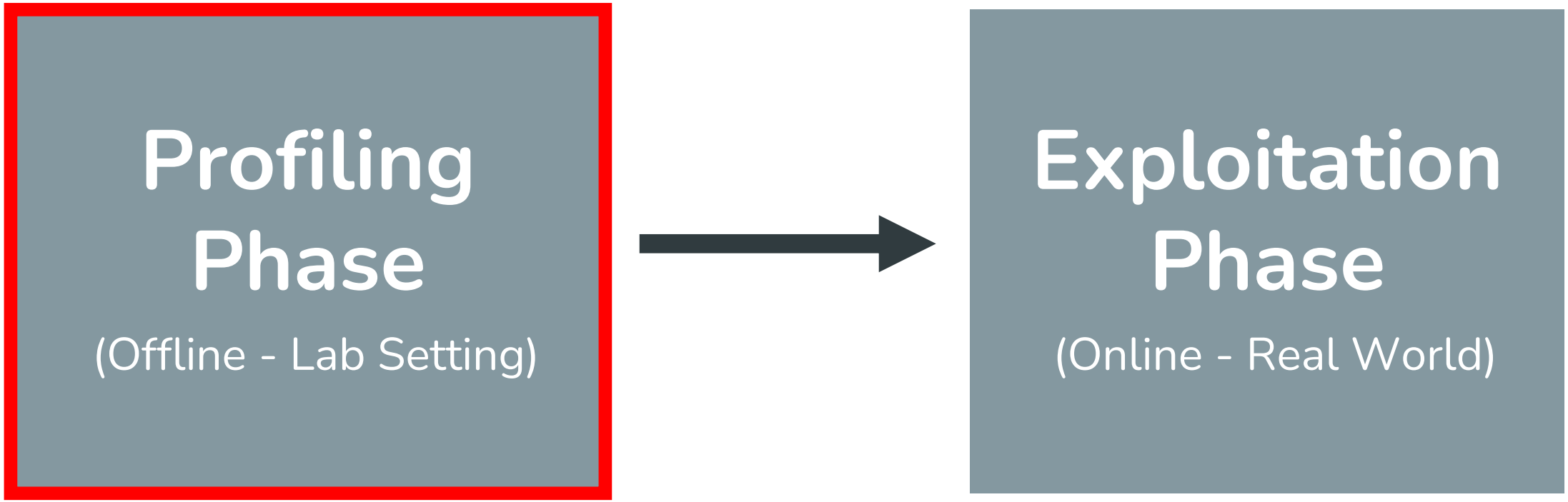
Different Contention Points

BUSted Profiling Phase

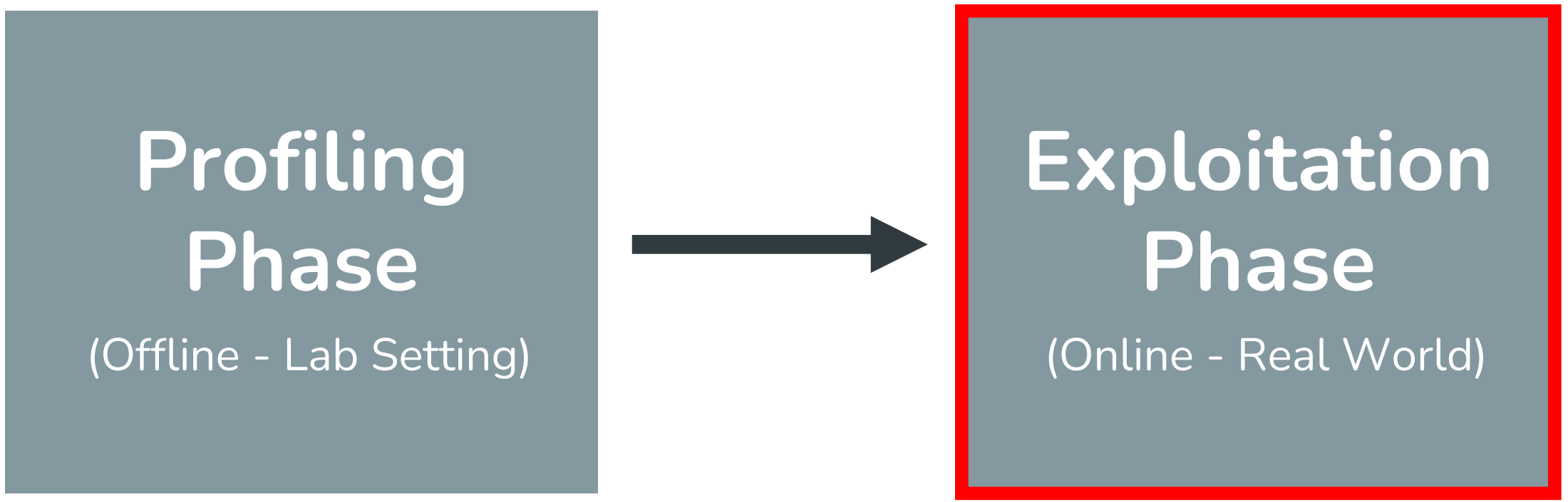


Let's Pick Contention Point 14

BUSted Attack Phases



BUSted Attack Phases



BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

for loop

BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

for loop

Contention in 14 ?

KEY = 0

BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

else branch

for loop

Contention in 14 ? NO

KEY = 0

BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

for loop

Contention in 14 ?

KEY = 1

BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

else branch

for loop

Contention in 14 ? NO

KEY = 1

BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

for loop

Contention in 14 ?

KEY = 2

BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

if branch (**leak**)

for loop

Contention in 14 ? YES

KEY = 2

BUSted Exploitation Phase

```
signed int read_keypad(void){
    int is_pressed, mask = 0x1;
    int new_key_state = get_keypad_state();
    for (int key = 0; key < KYPD_NB_KEYS; key++){
        is_pressed = (new_key_state & mask) & ~(key_state & mask);
        if (is_pressed)
          pin[pin_idx++] = key;
        else
          dummy_pin[dummy_pin_idx++] = key;
        dummy_pin_idx = 0;
        mask <<= 1;
    }
    key_state = new_key_state;
    return (4 - pin_idx);
}
void read_pin(){
  signed int pin_len = PIN_LEN;
  while(pin_len>0)
    pin_len = read_keypad();
}
```

if branch (**leak**)

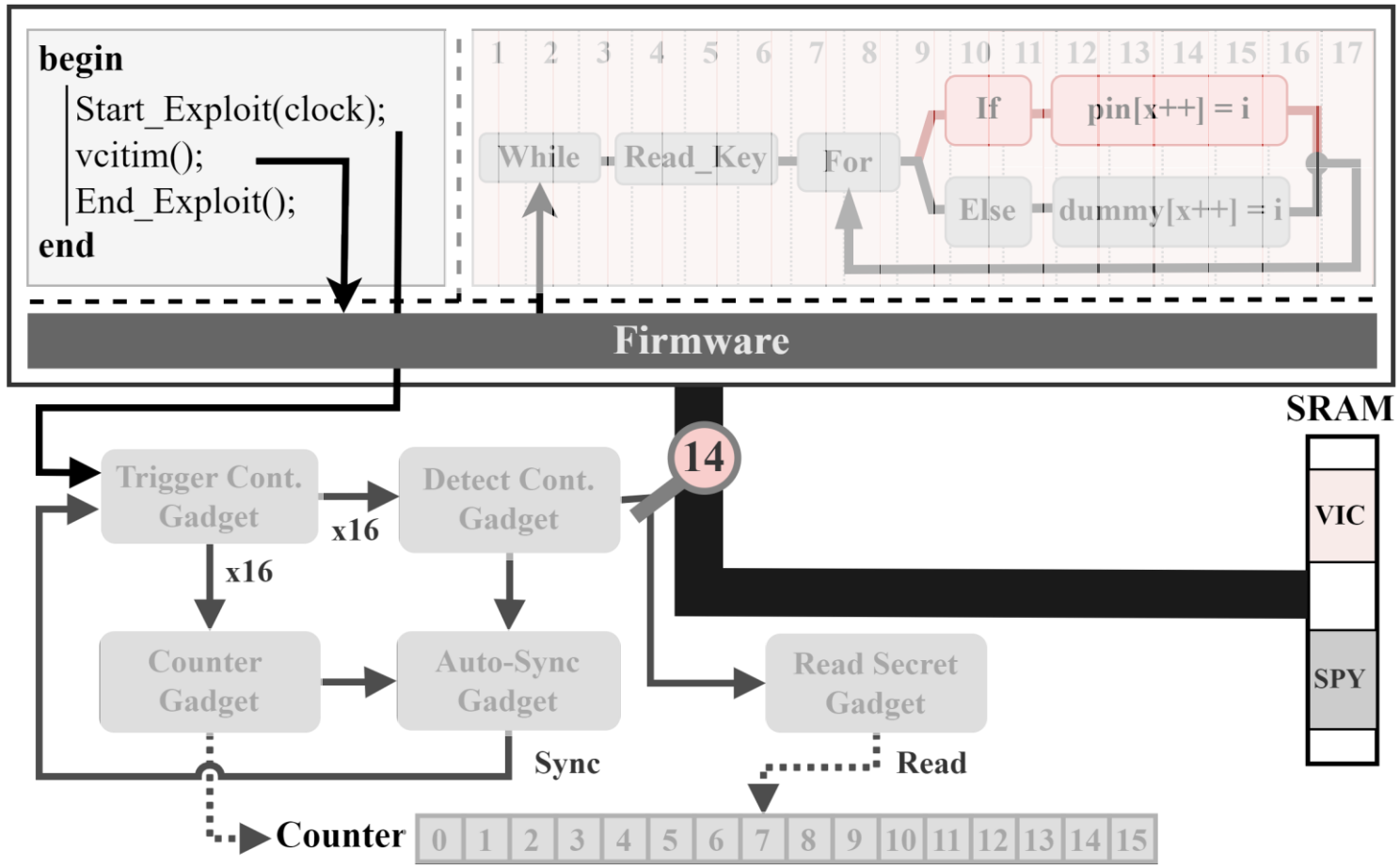
for loop

Contention in 14 ? YES

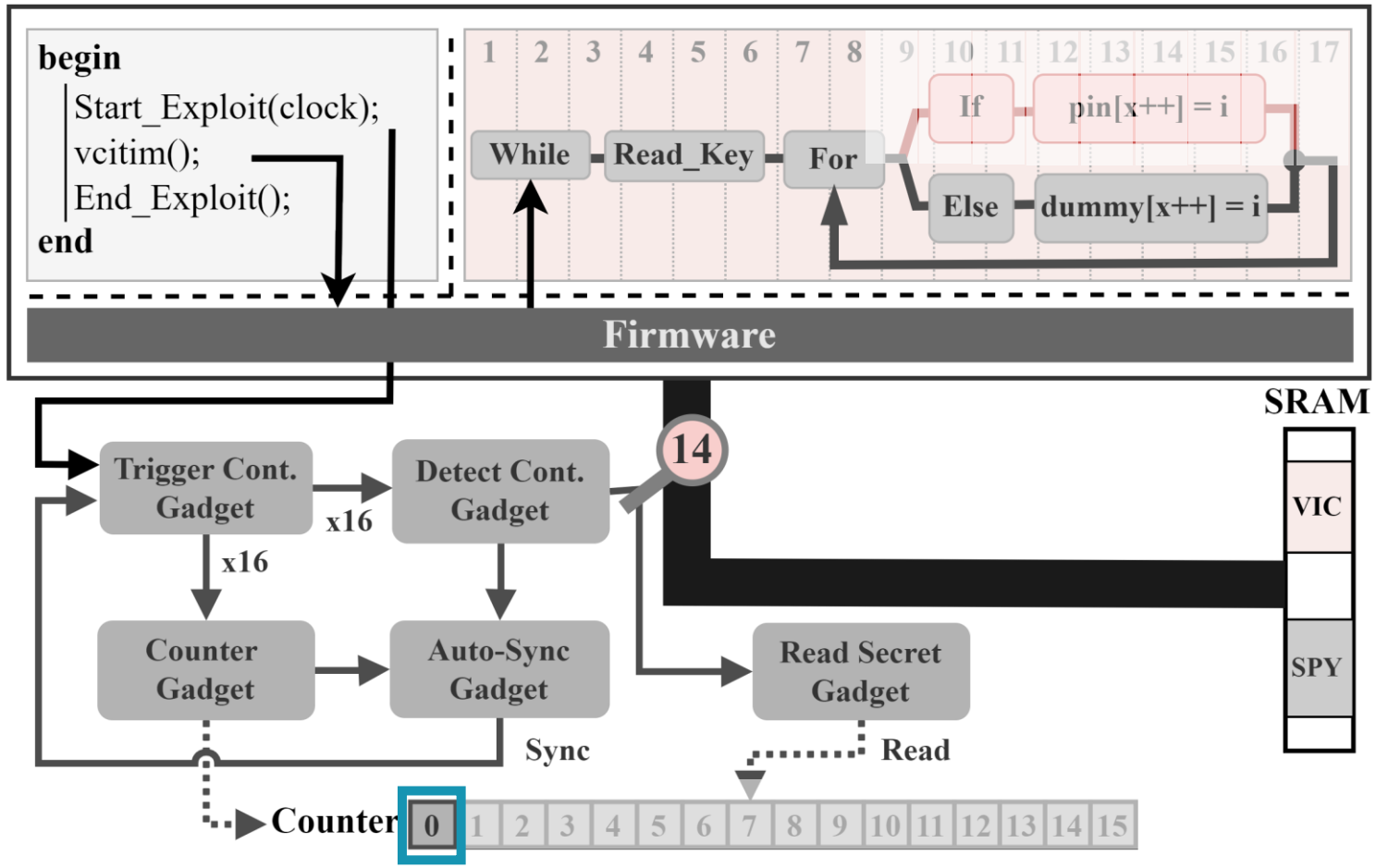
KEY = 2

Secret = 2

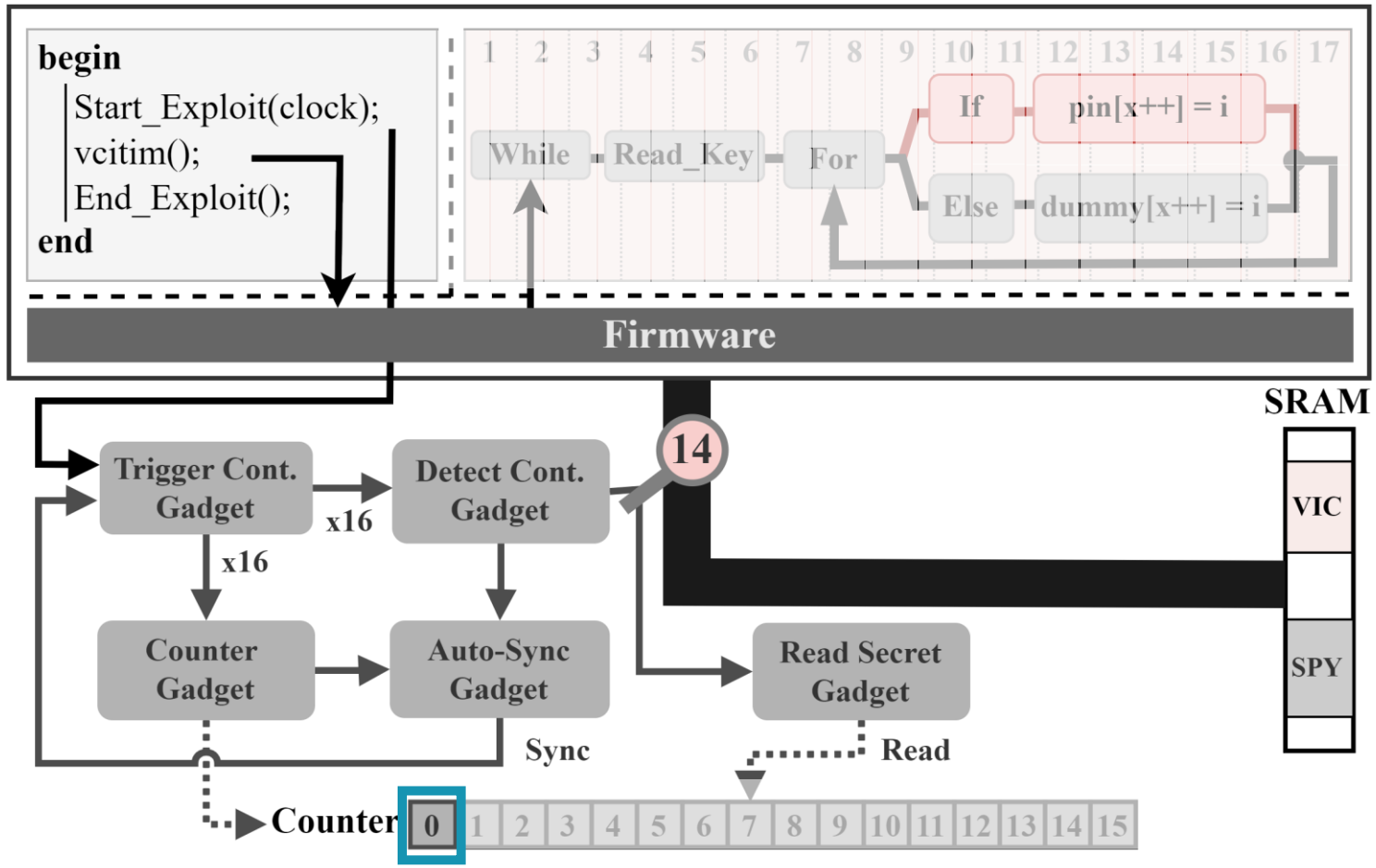
BUSted Exploitation Phase



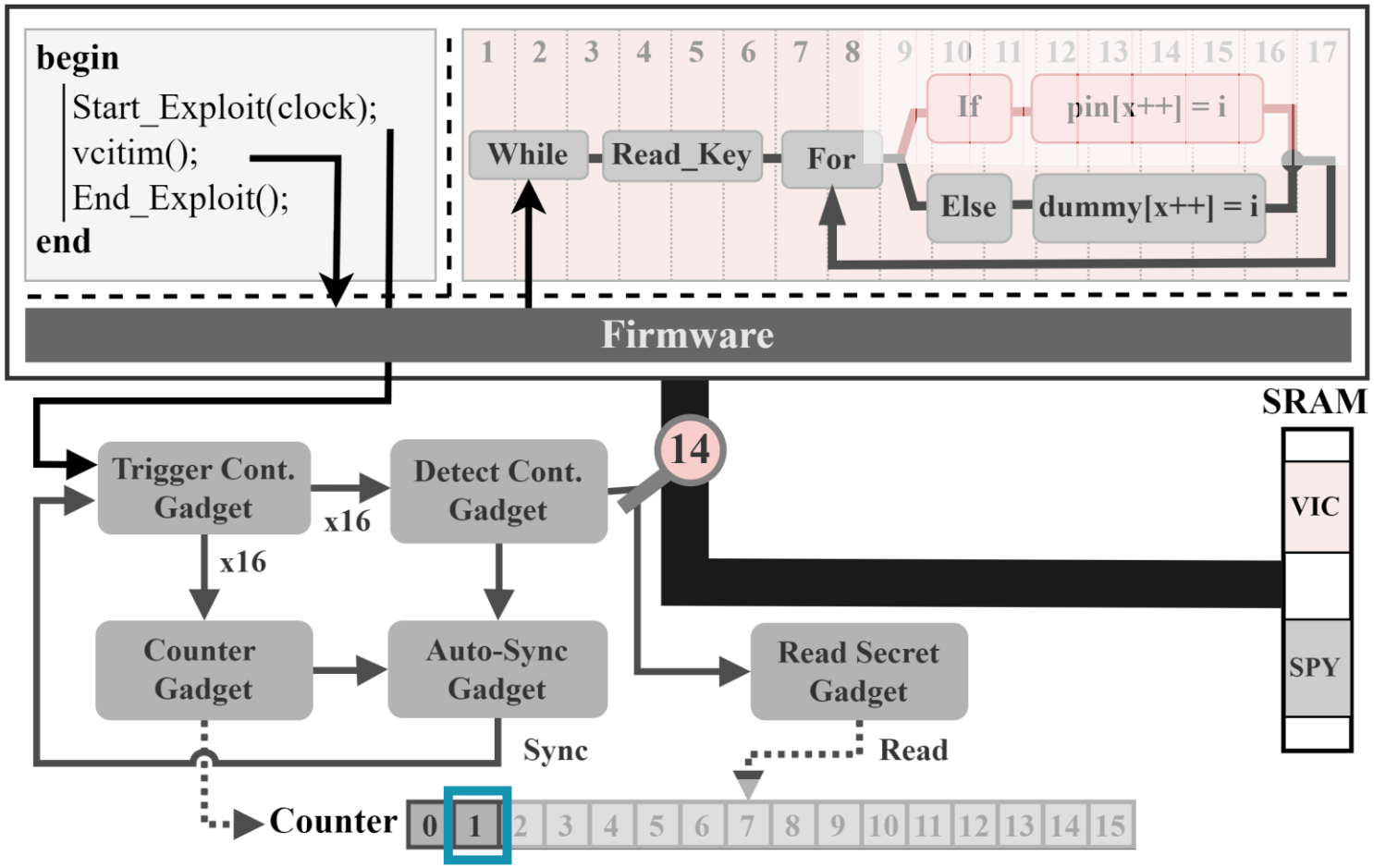
BUSted Exploitation Phase



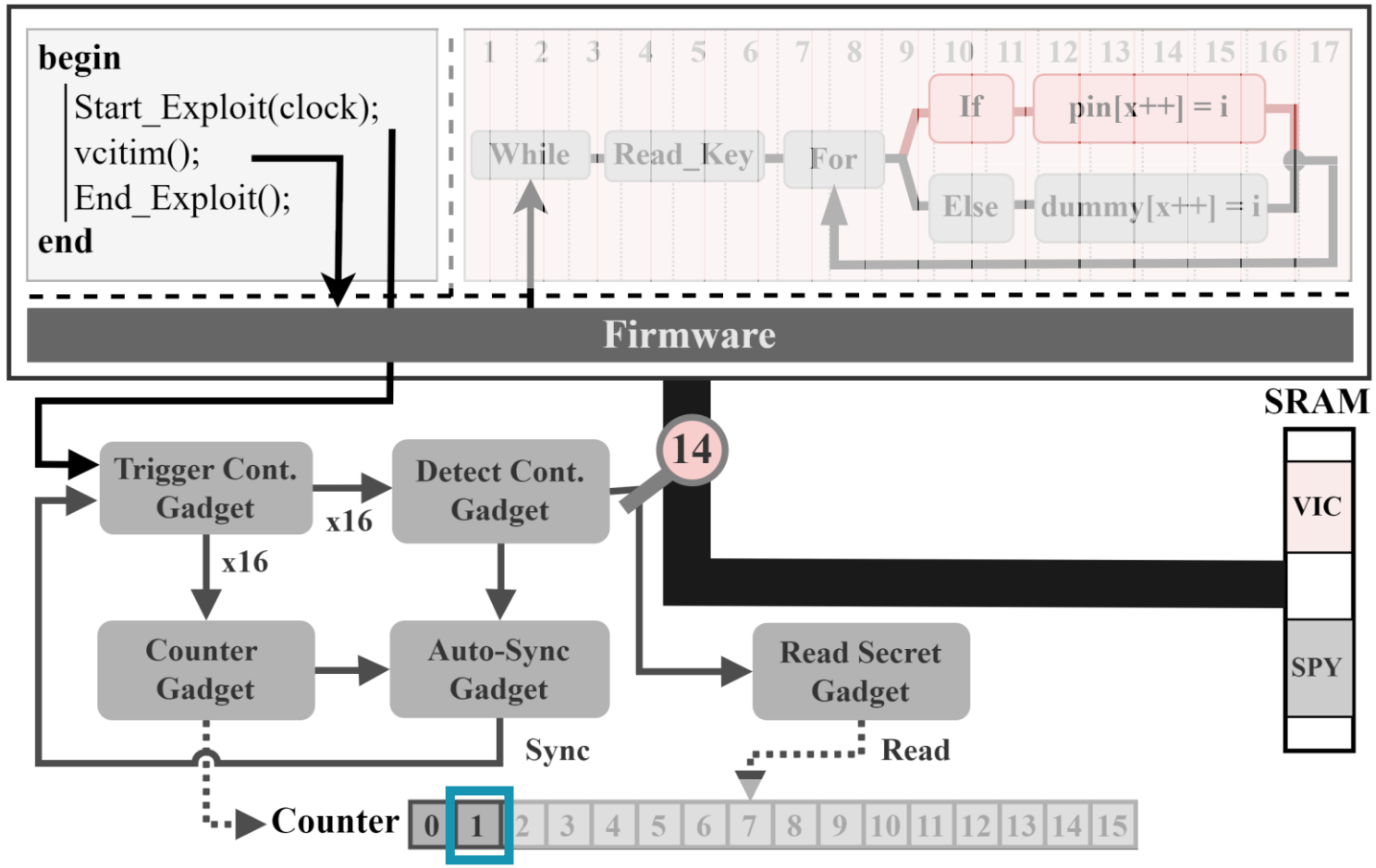
BUSted Exploitation Phase



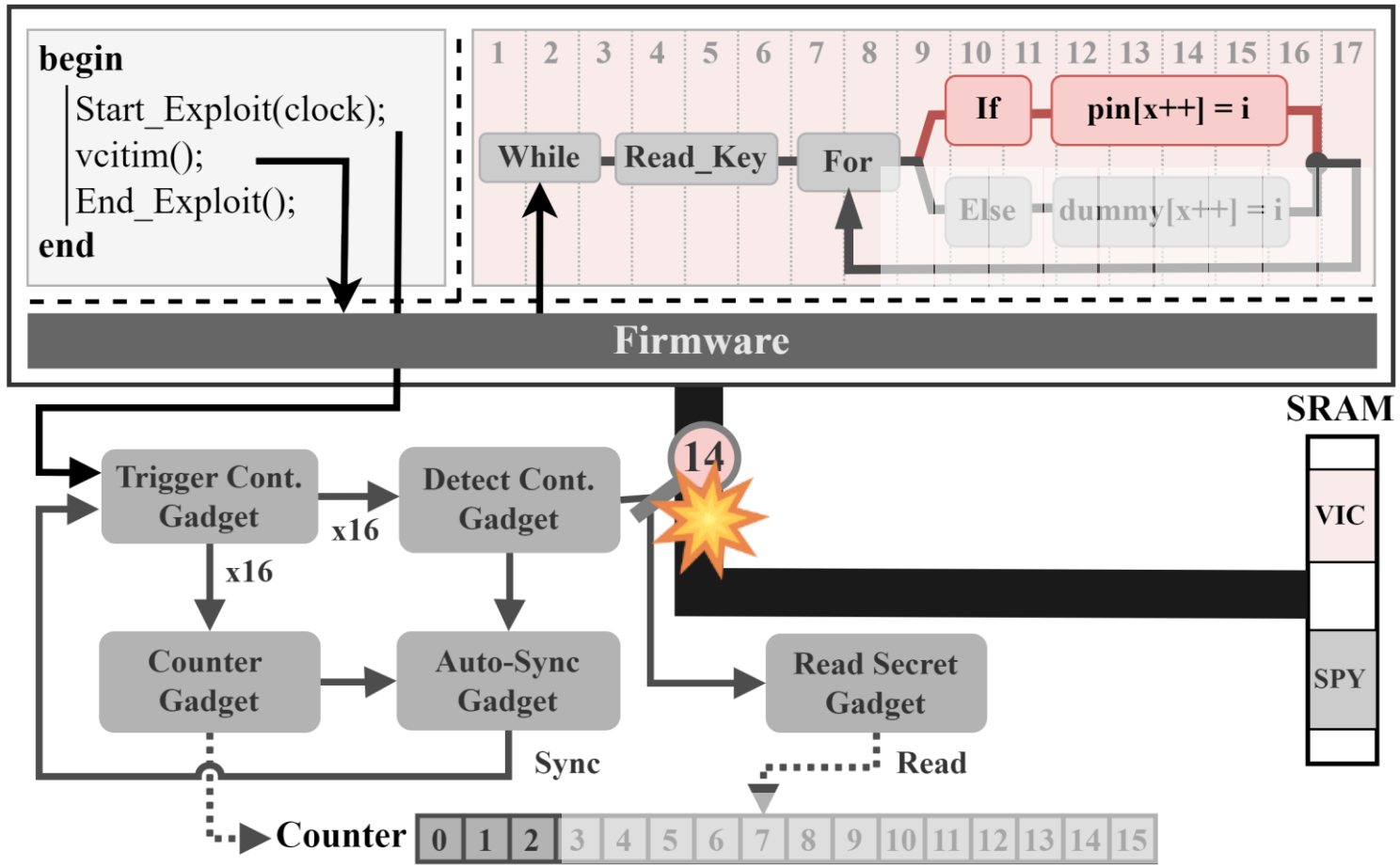
BUSted Exploitation Phase



BUSted Exploitation Phase

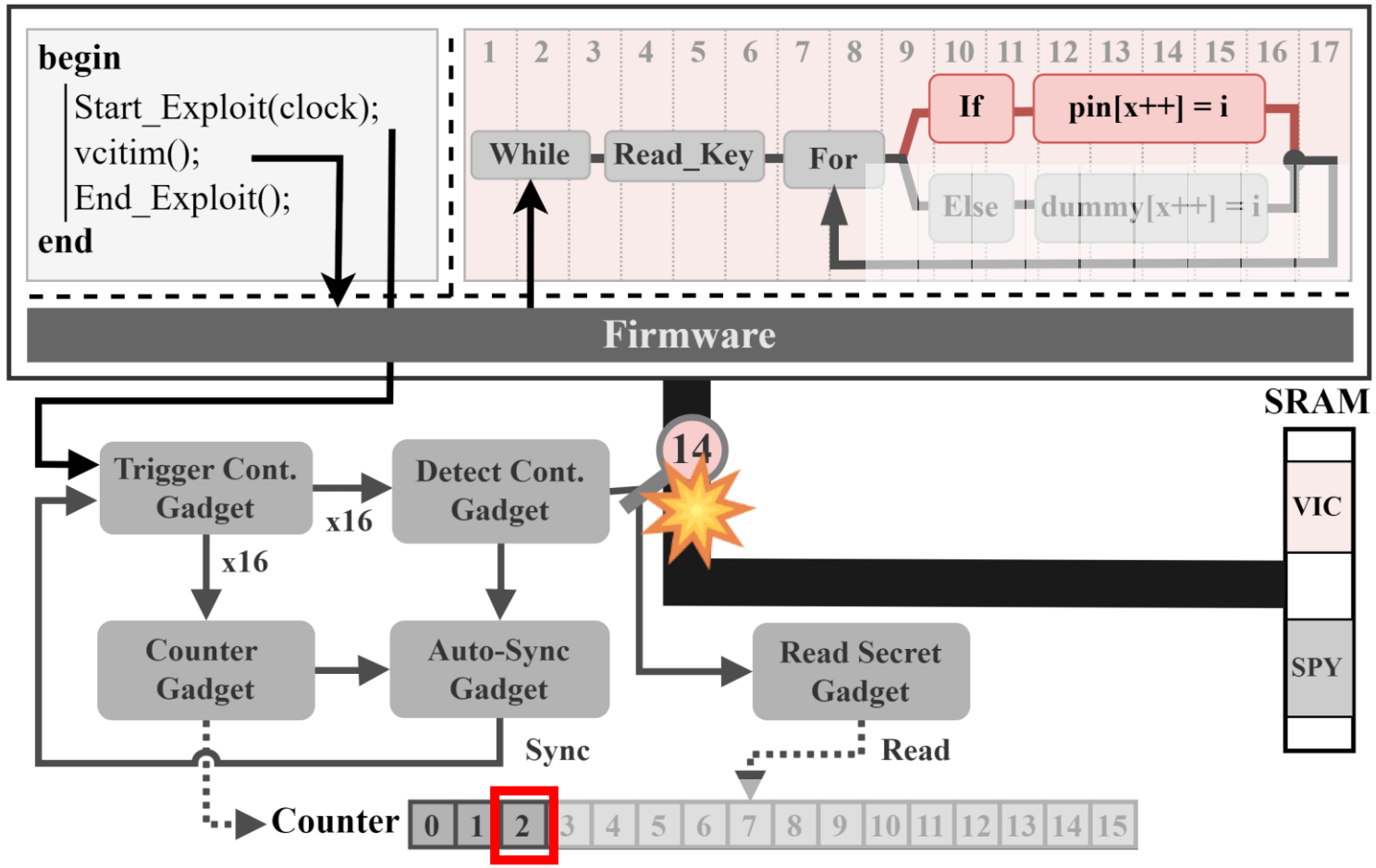


BUSted Exploitation Phase



CONTENTION!!

BUSted Exploitation Phase

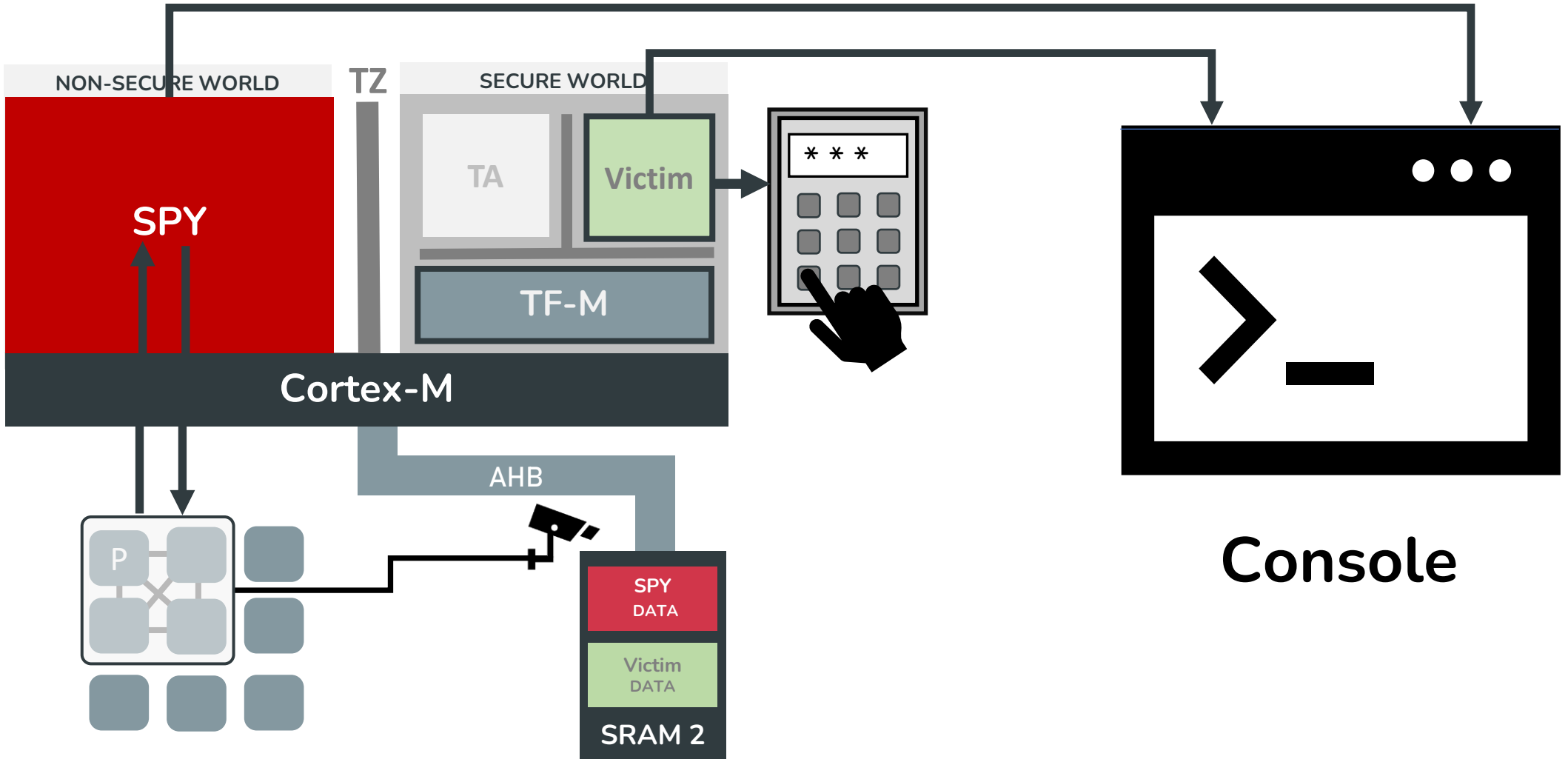


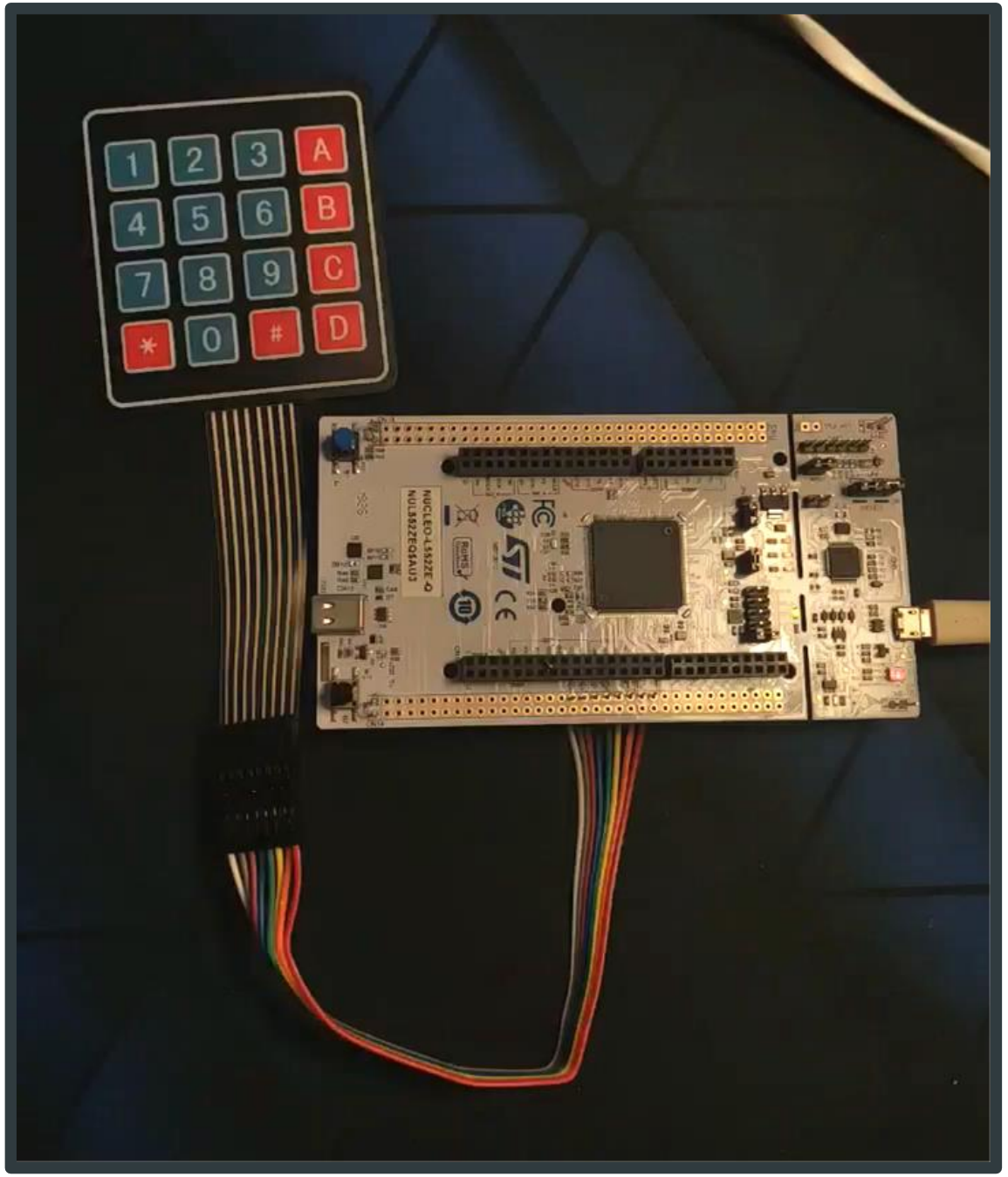
Secret = 2

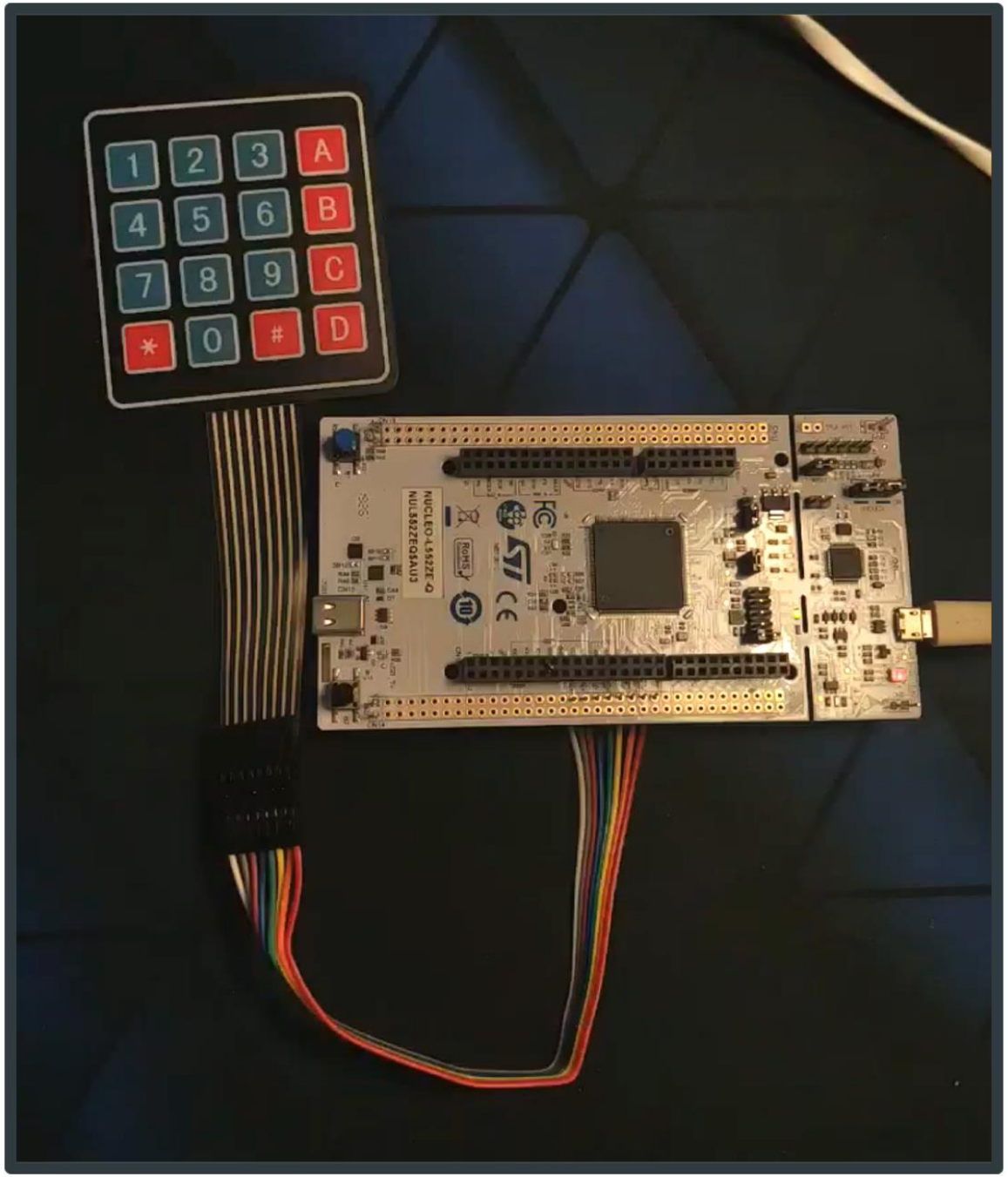
“Live” Demo

Demo of BUSted Attack

“Live” Demo Setup

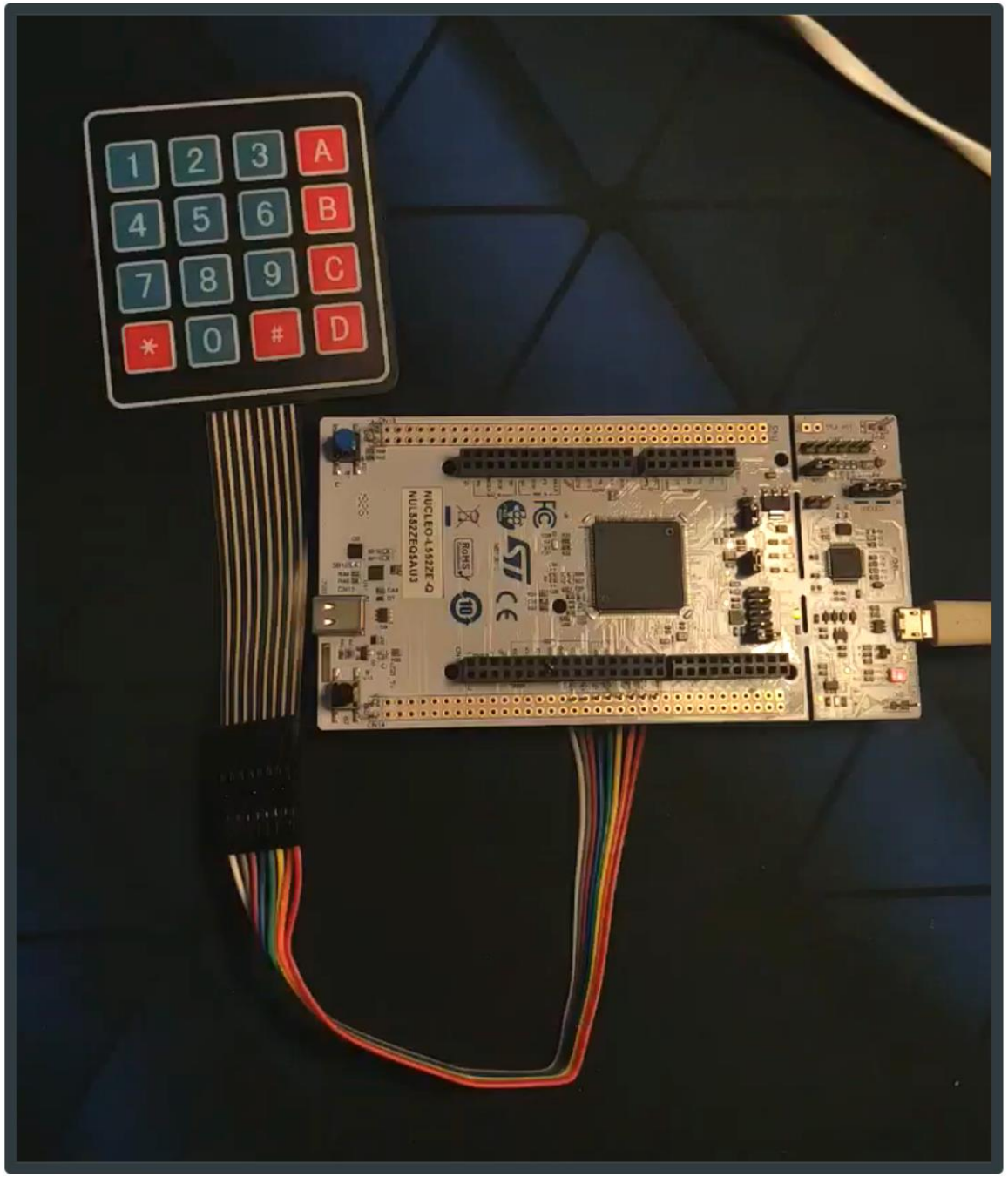



1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D

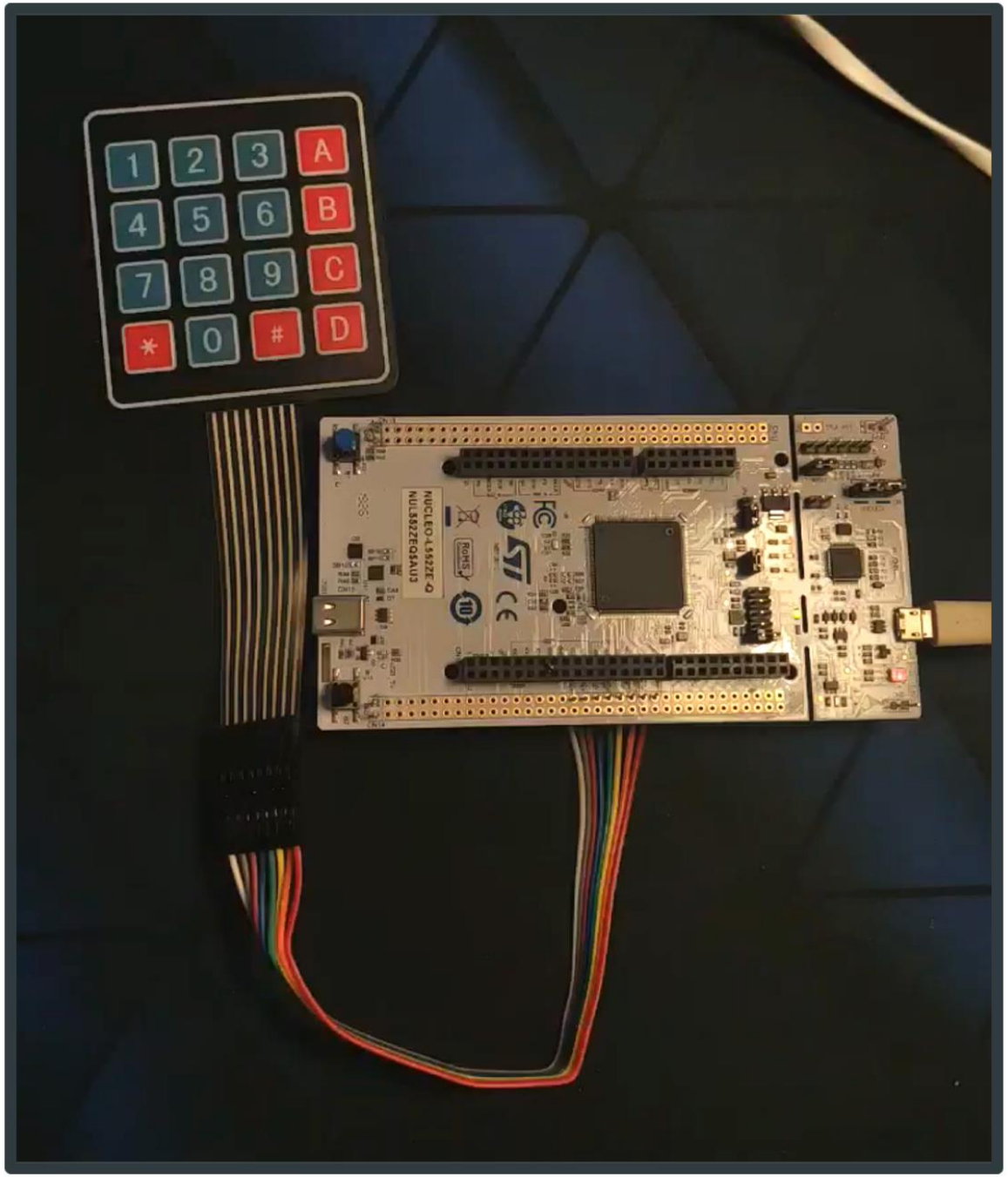



```
[INF] Starting bootloader
[INF] Swap type: none
[INF] Swap type: none
[INF] Bootloader chainload address offset: 0x13000
[INF] Jumping to the first image slot
[Sec Thread] Secure image initializing!
TF-M isolation level is: 0x00000002
Booting TFM v1.4.0

[NS] Non-Secure system starting
```

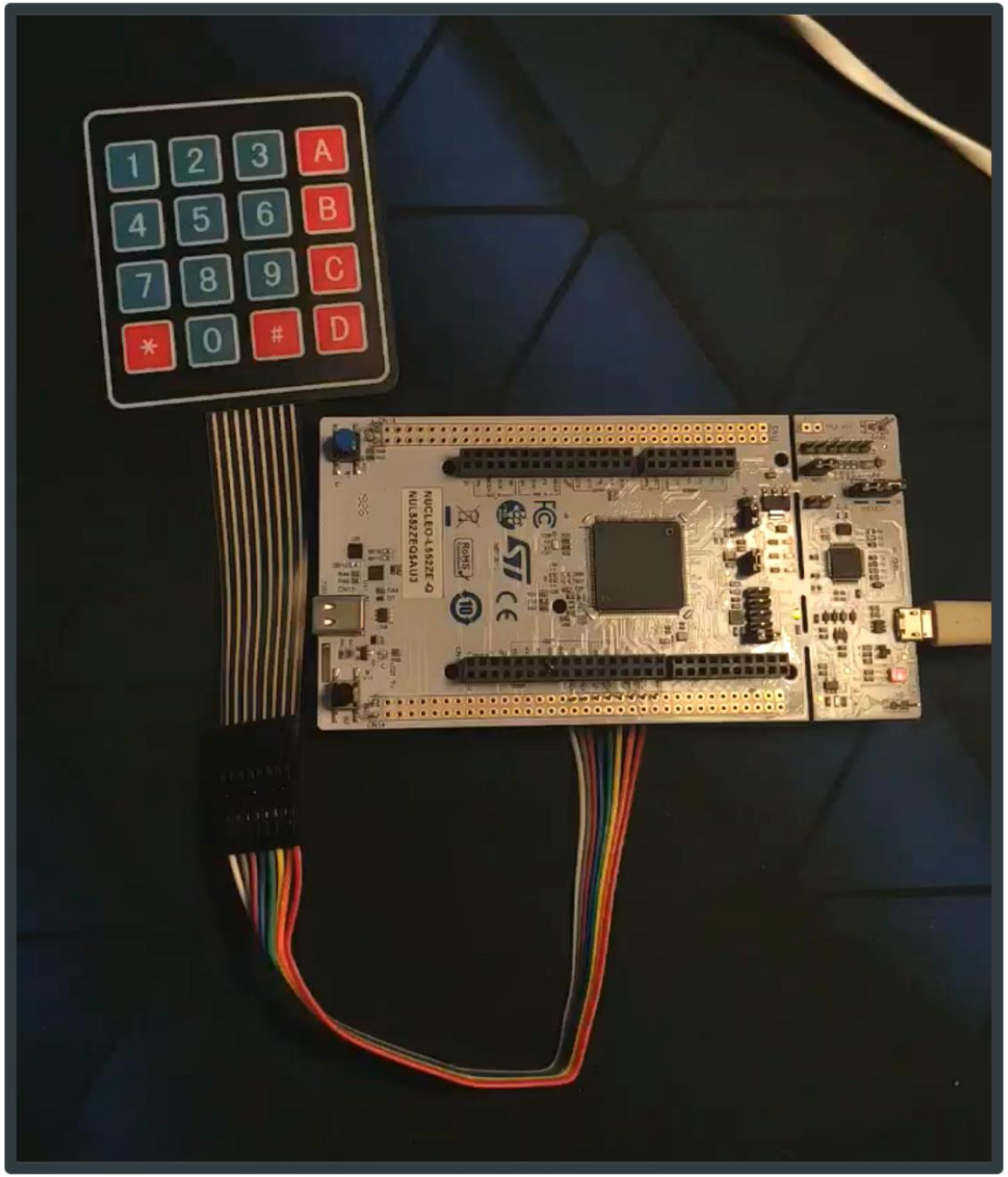
```
*****************************************
||                  SPY                ||
*****************************************
[NS] Hardware Gadgets Are Configured
[NS] SPY Is Running
[NS] Invoking Smart Lock TA

     ************************************
     ||        SMART LOCK TA           ||
     ************************************
     [TA] Reading Trusted Keypad
     [TA] Secure PIN =  7 5 8 6
     [TA] Leaving SMART LOCK TA
     ************************************
     ************************************

[NS] Stolen PIN  = 7 5 8 6
*****************************************
*****************************************
```



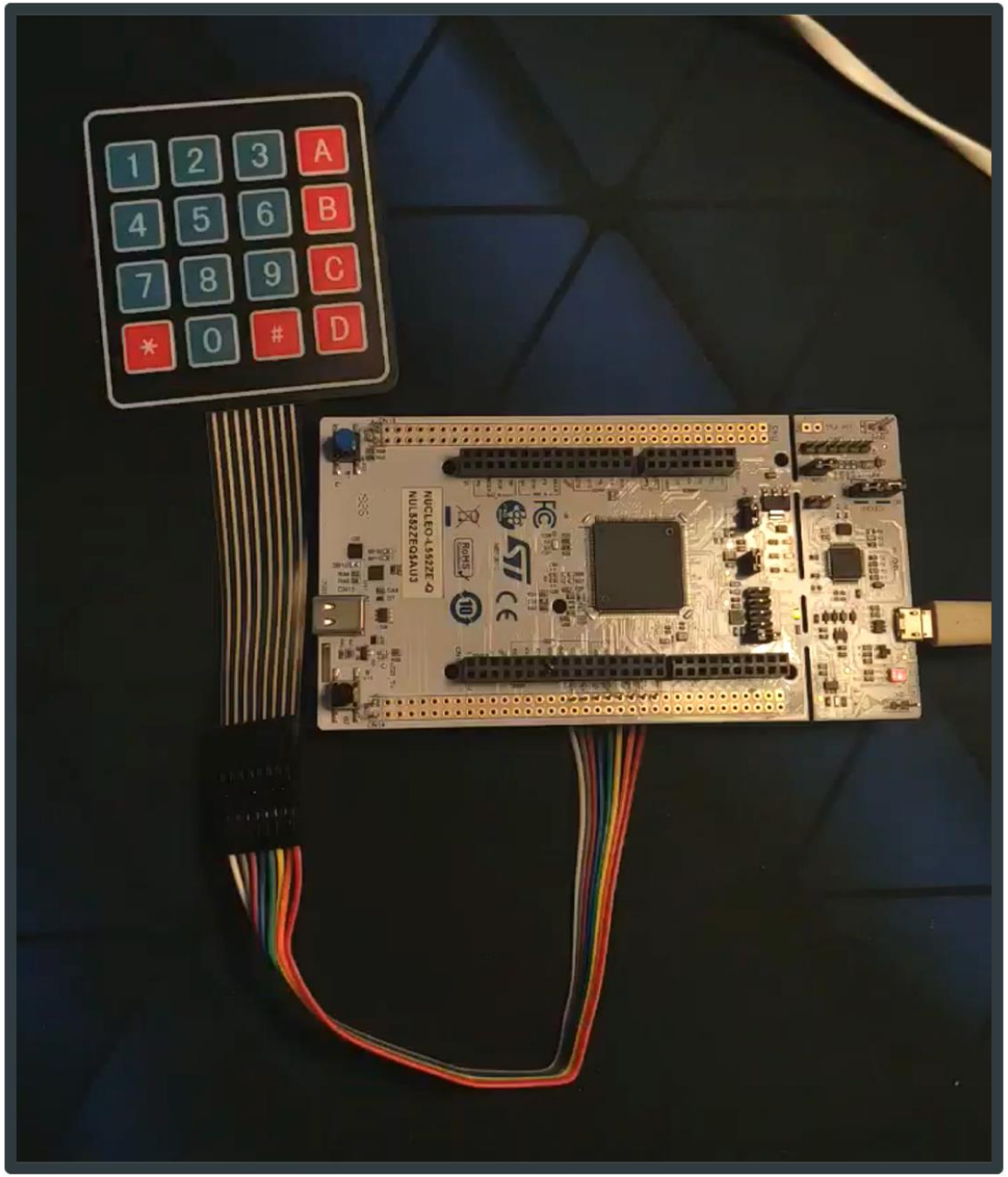

```
[INF] Starting bootloader
[INF] Swap type: none
[INF] Swap type: none
[INF] Bootloader chainload address offset: 0x13000
[INF] Jumping to the first image slot
[Sec Thread] Secure image initializing!
TF-M isolation level is: 0x00000002
Booting TFM v1.4.0

[NS] Non-Secure system starting

*****************************************
||                  SPY                ||
*****************************************
[NS] Hardware Gadgets Are Configured
[NS] SPY Is Running
[NS] Invoking Smart Lock TA

     ************************************
     ||        SMART LOCK TA           ||
     ************************************
     [TA] Reading Trusted Keypad
     [TA] Secure PIN =  7 5 8 6
     [TA] Leaving SMART LOCK TA
     ************************************
     ************************************

[NS] Stolen PIN  = 7 5 8 6
*****************************************
*****************************************
```




```
[INF] Starting bootloader
[INF] Swap type: none
[INF] Swap type: none
[INF] Bootloader chainload address offset: 0x13000
[INF] Jumping to the first image slot
[Sec Thread] Secure image initializing!
TF-M isolation level is: 0x00000002
Booting TFM v1.4.0

[NS] Non-Secure system starting

*****************************************
||                  SPY                ||
*****************************************
[NS] Hardware Gadgets Are Configured
[NS] SPY Is Running
[NS] Invoking Smart Lock TA

    *************************************
    ||        SMART LOCK TA            ||
    *************************************
    [TA] Reading Trusted Keypad
    [TA] Secure PIN =  7 5 8 6
    [TA] Leaving SMART LOCK TA
    *************************************
    *************************************

[NS] Stolen PIN  = 7 5 8 6
*****************************************
*****************************************
```




```
[INF] Starting bootloader
[INF] Swap type: none
[INF] Swap type: none
[INF] Bootloader chainload address offset: 0x13000
[INF] Jumping to the first image slot
[Sec Thread] Secure image initializing!
TF-M isolation level is: 0x00000002
Booting TFM v1.4.0

[NS] Non-Secure system starting

*****************************************
||                  SPY                ||
*****************************************
[NS] Hardware Gadgets Are Configured
[NS] SPY Is Running
[NS] Invoking Smart Lock TA

     *************************************
     ||        SMART LOCK TA            ||
     *************************************
     [TA] Secure PIN =  7 5 8 6
     [TA] Leaving SMART LOCK TA
     *************************************
     *************************************

[NS] Stolen PIN  = 7 5 8 6
*****************************************
*****************************************
```




```
[INF] Starting bootloader
[INF] Swap type: none
[INF] Swap type: none
[INF] Bootloader chainload address offset: 0x13000
[INF] Jumping to the first image slot
[Sec Thread] Secure image initializing!
TF-M isolation level is: 0x00000002
Booting TFM v1.4.0

[NS] Non-Secure system starting

*****************************************
||                  SPY                ||
*****************************************
[NS] Hardware Gadgets Are Configured
[NS] SPY Is Running
[NS] Invoking Smart Lock TA

     *************************************
     ||        SMART LOCK TA            ||
     *************************************
     [TA] Secure PIN =  7 5 8 6
     [TA] Leaving SMART LOCK TA
     *************************************
     *************************************

[NS] Stolen PIN  = 7 5 8 6
*****************************************
*****************************************
```

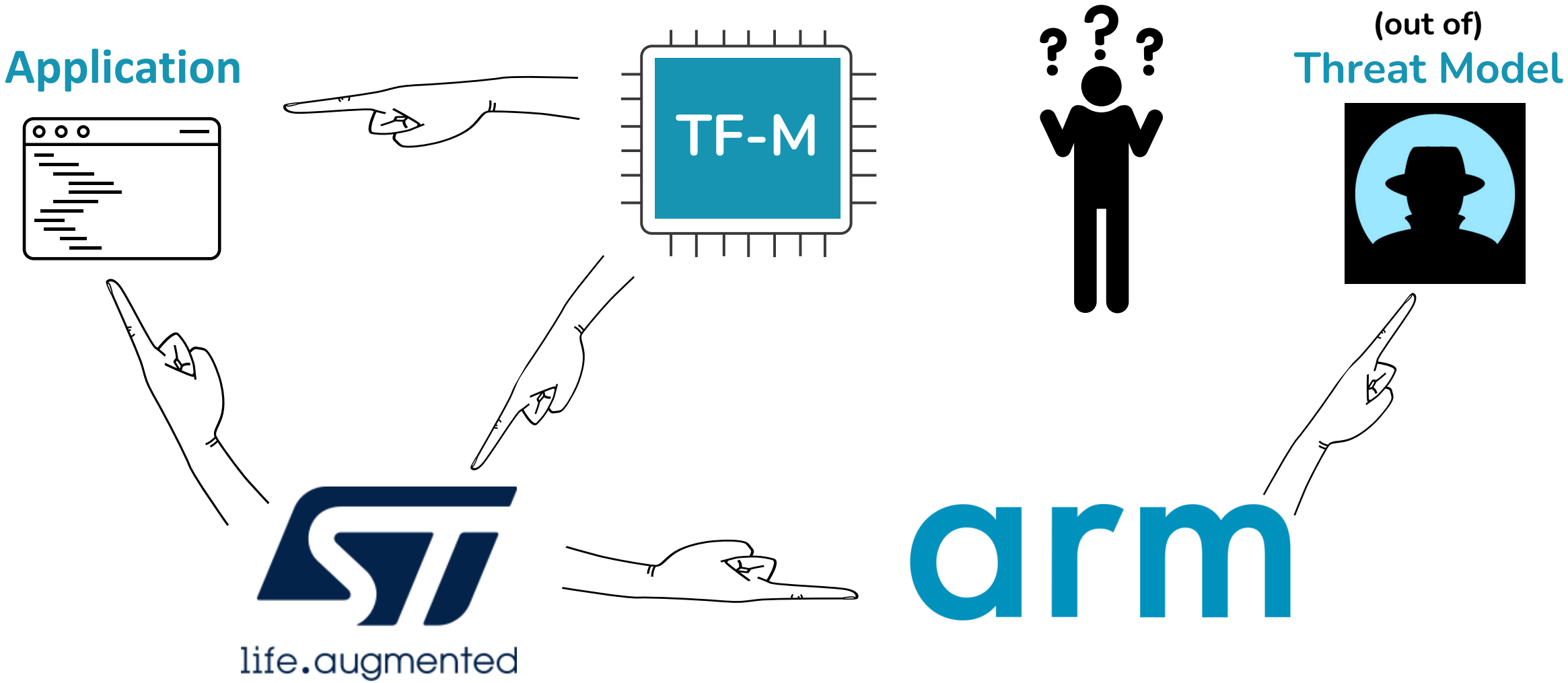
Summary

Responsible Disclosure, and Black Hat Sound Bytes

We presented **BUSted**, the **1st microarchitectural side-channel attack** for **TrustZone-M** MCUs.

Responsible Disclosure

Responsible Disclosure



Black Hat

SOUND BYTES

1. Evidence that **MCUs** are **vulnerable** to **microarchitectural** side-channel **attacks**;

2. **New class** of software-based **microarchitectural** timing side-channel **attacks** affecting **MCUs**;

3. **Reference attack** that bypasses the TEE of state-of-the-art **TrustZone-M MCUs**!

THANK YOU!

Cristiano Rodrigues | Sandro Pinto, PhD
(Centro ALGORITMI / LASI, Universidade do Minho)

id9492@alunos.uminho.pt

 **@_CRodrigues__**

sandro.pinto@dei.uminho.pt

 **@sandro2pinto**

Q&A

Backup Slides

Countermeasures

1. Avoid secret-dependent code;
2. Avoid sharing memory banks;
3. Disable DMA during the victim execution;
4. Enforce priority-based arbitration policy:
5. Add random delays to the victim code.