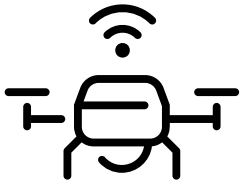
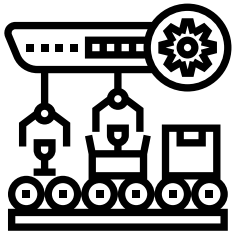
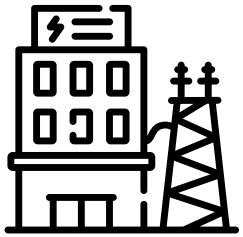
Scheduling Obfuscation: Analyzing the Endeavor of Deterring Timing Inference Attacks on Real-Time Systems

05.08.2020

Simon Langer

Friedrich-Alexander-Universität Erlangen-Nürnberg

Beispiele für Echtzeitsysteme (Real-Time Systems)

- 
  Airbag, ABS
- 
  
  Flugsysteme
- 
  
  Anlagensteuerung
- 
  Diagnose, Behandlung

Icons made by Eucalyp, monkik, mavadee, Vitaly Gorbachev, Freepik from www.flaticon.com

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP

Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvBK/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP

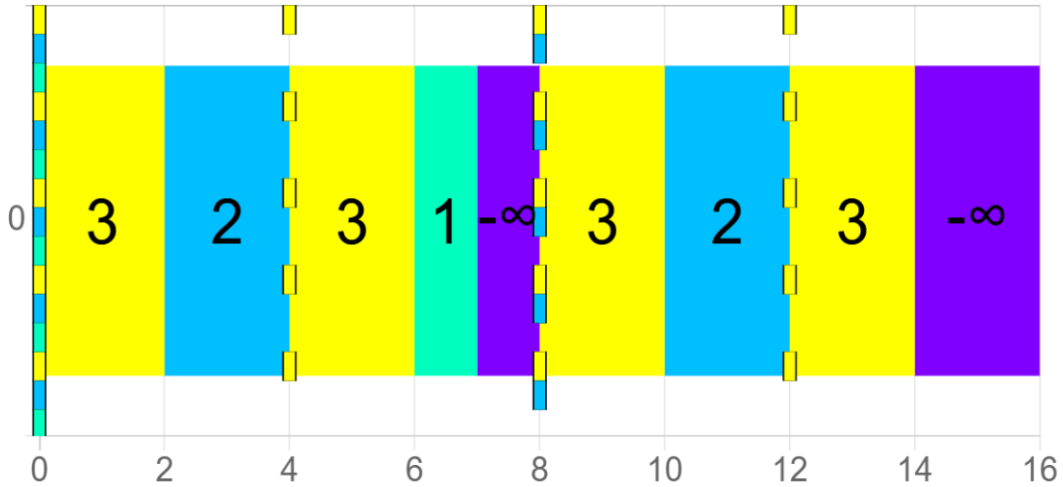
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvBK/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork
2	0	4	4	3	0
2	0	8	8	2	0
1	0	16	16	1	0
Infinity	0	Infinity	Infinity	-Infinity	Infinity

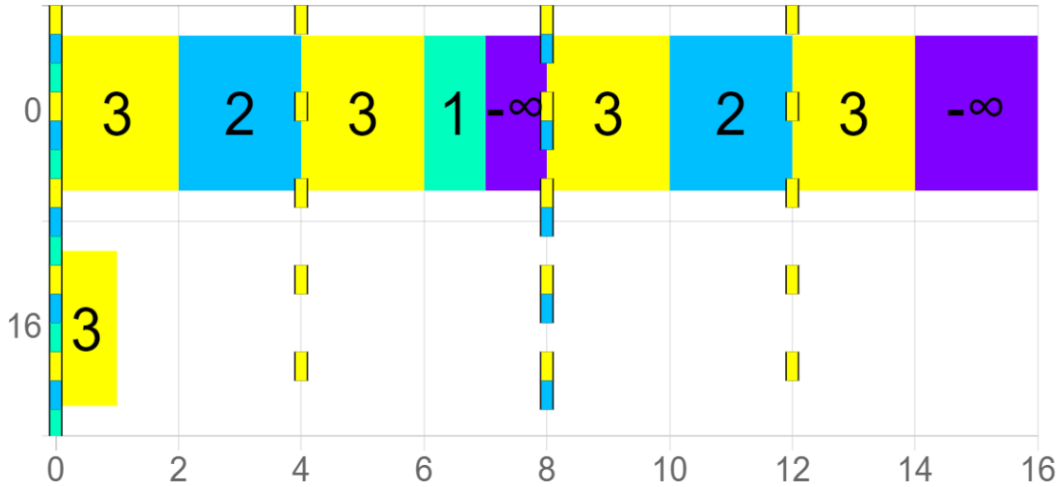
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	1	−
2	0	8	8	2	2	−
1	0	16	16	1	1	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

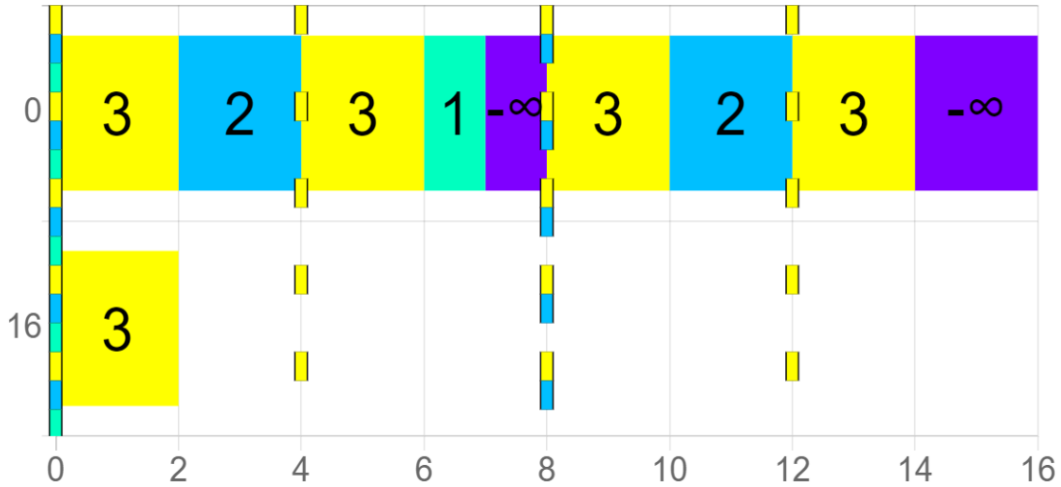
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	
2	0	4	4	3	0	
2	0	8	8	2	2	
1	0	16	16	1	1	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	

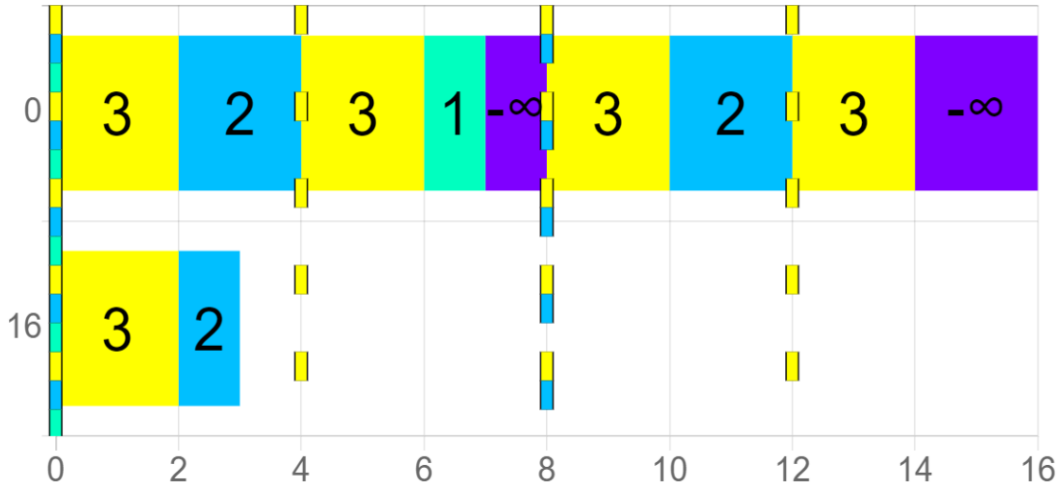
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	0	−
2	0	8	8	2	1	−
1	0	16	16	1	1	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

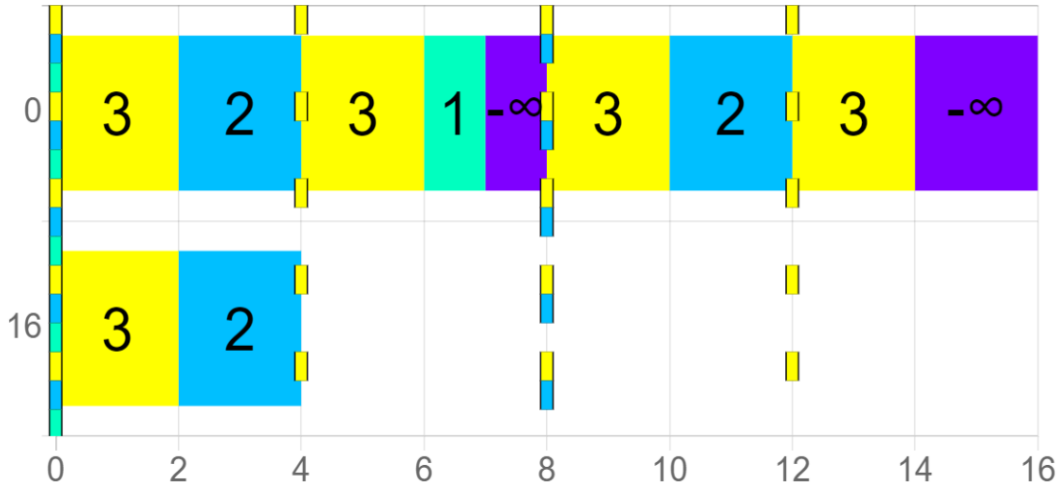
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	
2	0	4	4	3	0	
2	0	8	8	2	0	
1	0	16	16	1	1	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	

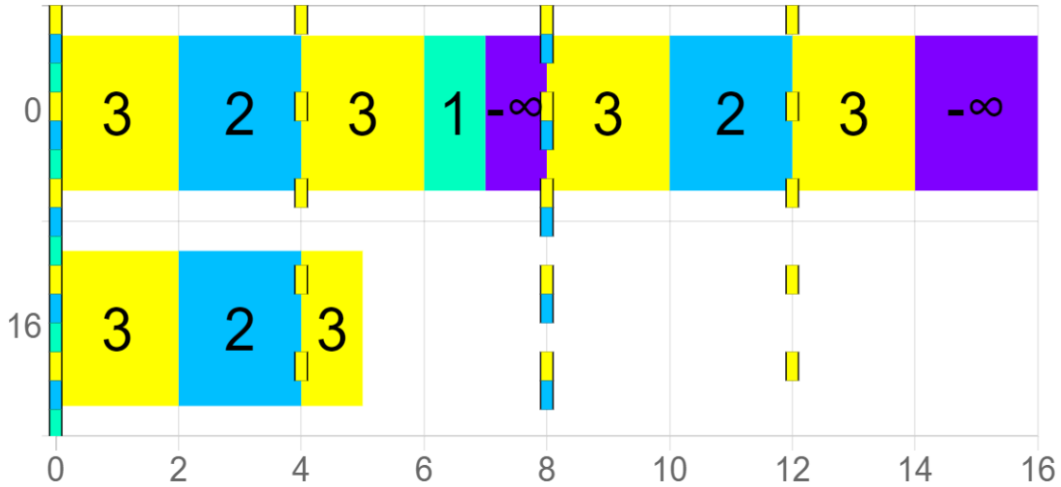
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	
2	0	4	4	3	1	
2	0	8	8	2	0	
1	0	16	16	1	1	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	

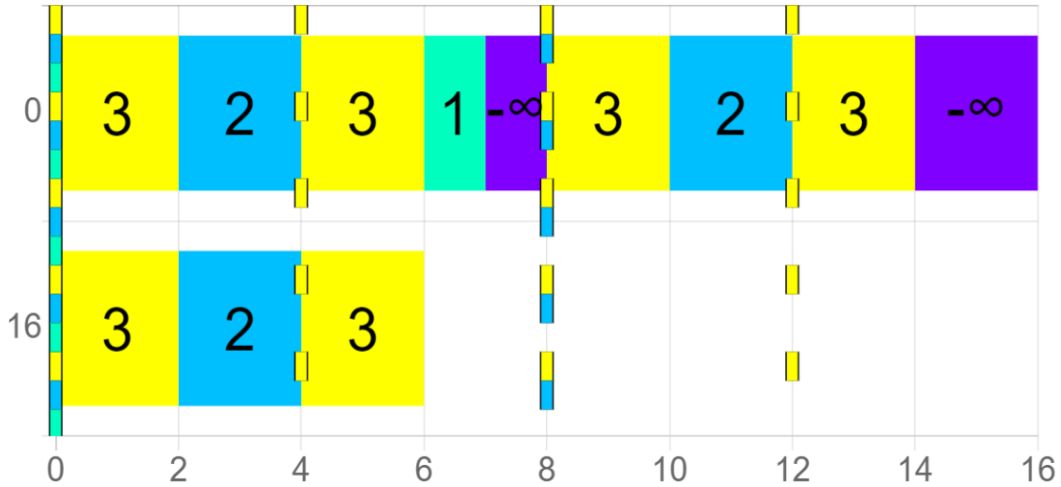
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	0	−
2	0	8	8	2	0	−
1	0	16	16	1	1	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

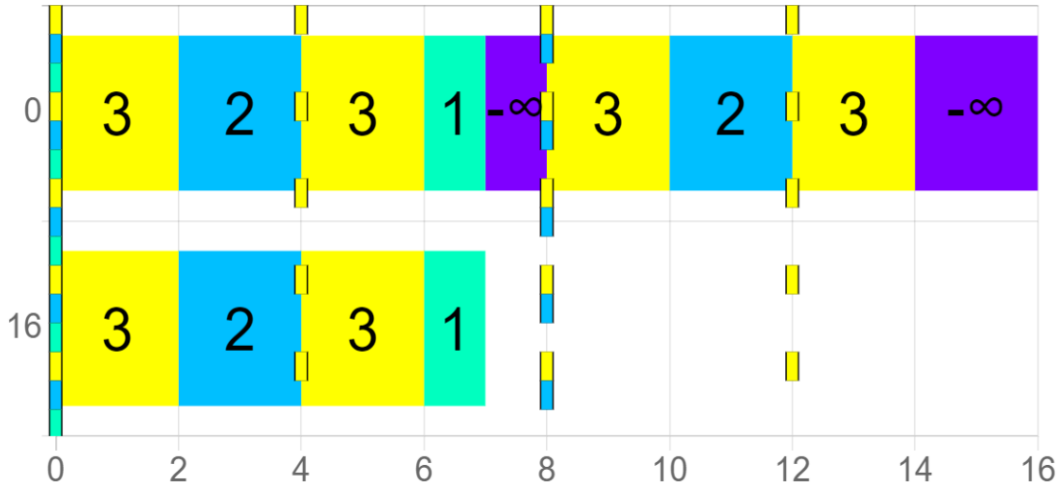
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	0	−
2	0	8	8	2	0	−
1	0	16	16	1	0	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

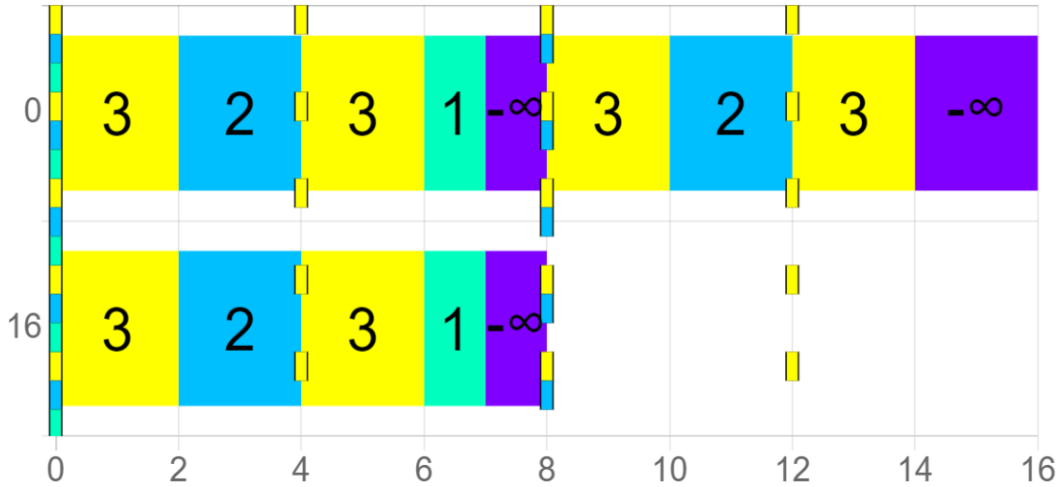
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	0	−
2	0	8	8	2	0	−
1	0	16	16	1	0	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

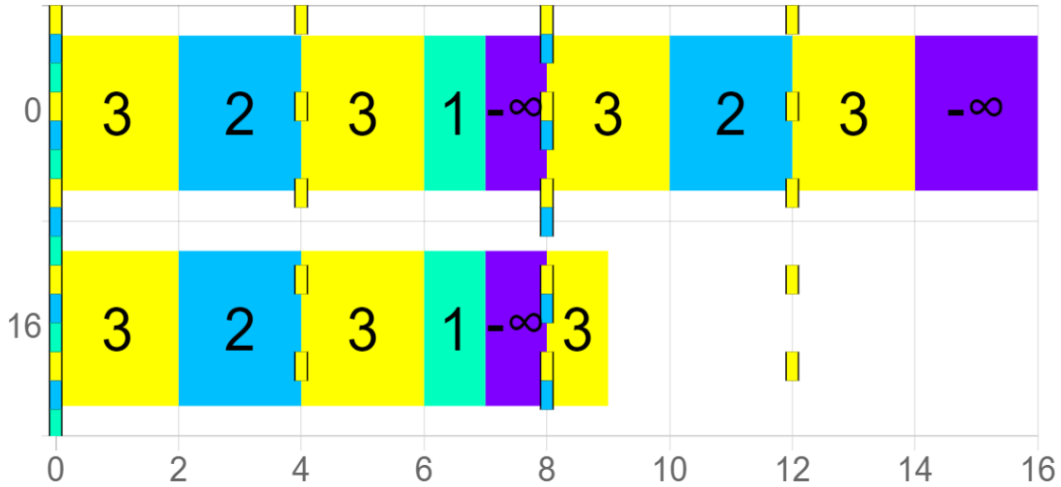
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	1	−
2	0	8	8	2	2	−
1	0	16	16	1	0	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

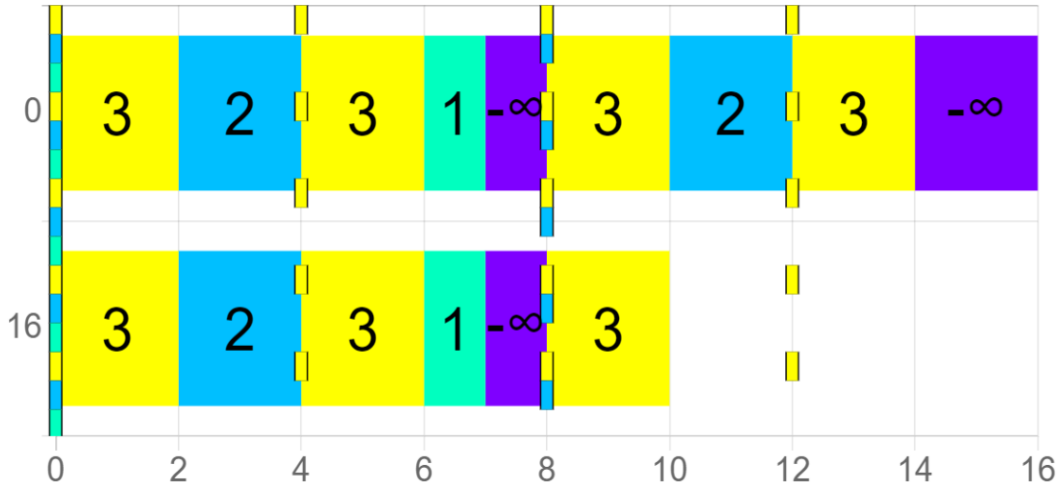
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	0	−
2	0	8	8	2	2	−
1	0	16	16	1	0	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

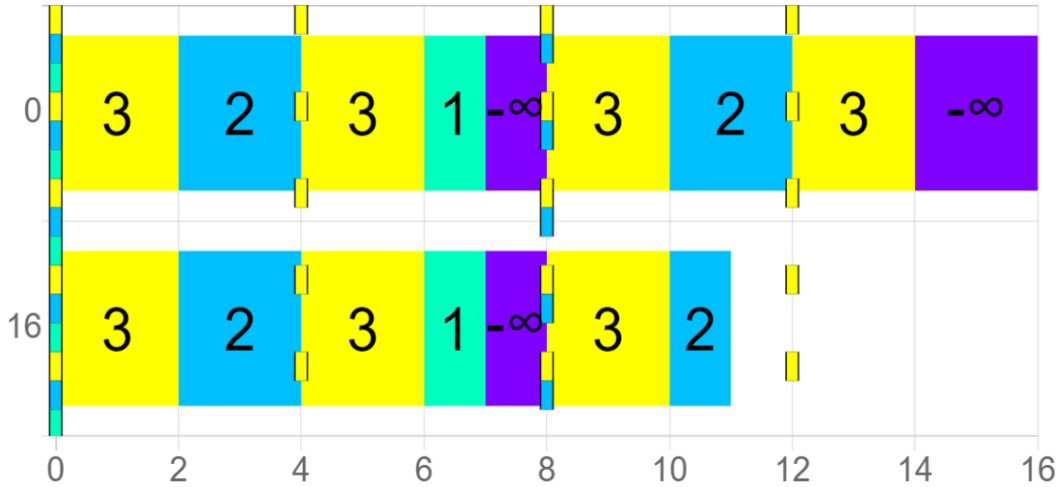
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	0	−
2	0	8	8	2	1	−
1	0	16	16	1	0	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

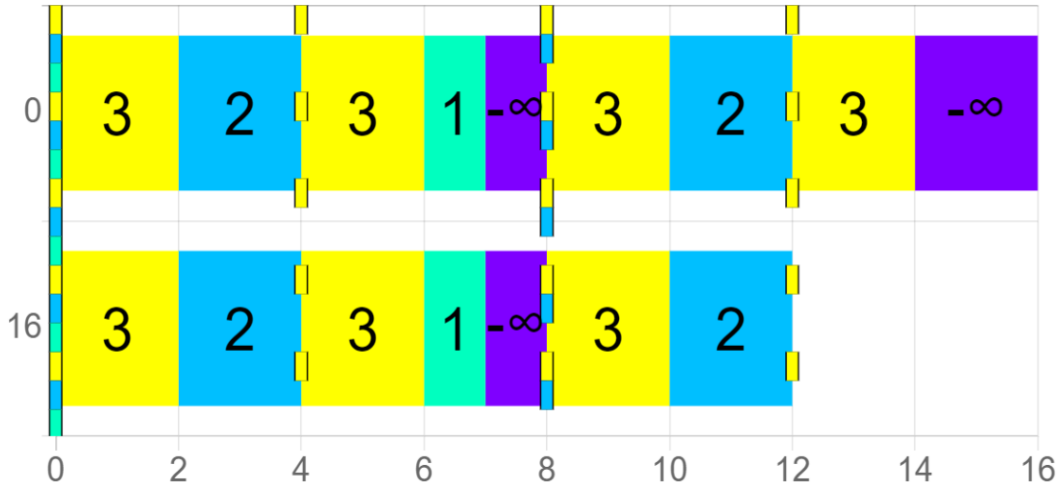
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	+
2	0	4	4	3	0	−
2	0	8	8	2	0	−
1	0	16	16	1	0	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	−

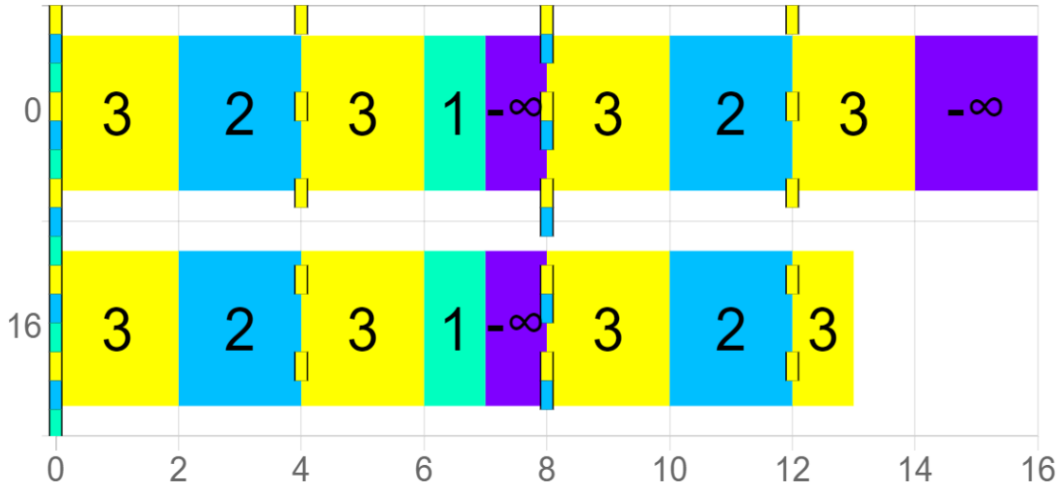
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	
2	0	4	4	3	1	
2	0	8	8	2	0	
1	0	16	16	1	0	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	

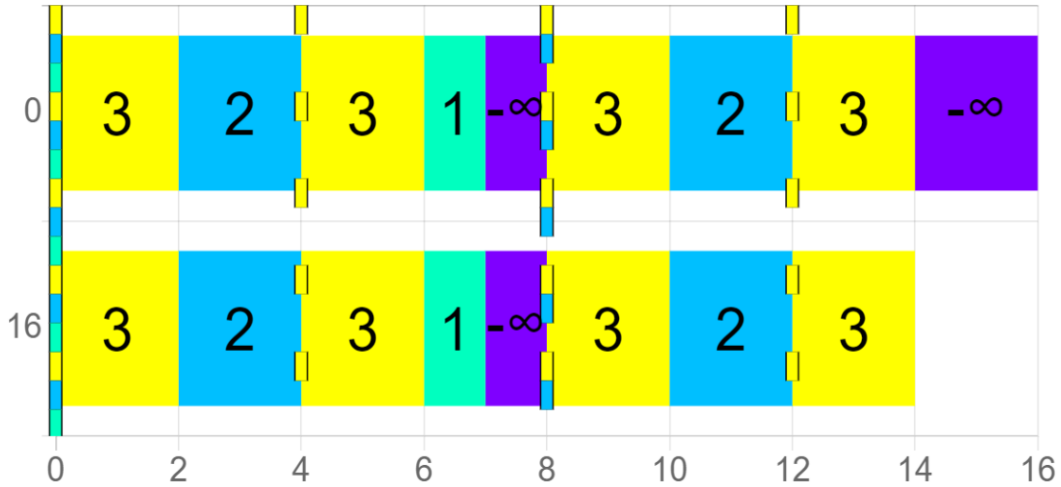
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	
2	0	4	4	3	0	
2	0	8	8	2	0	
1	0	16	16	1	0	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	

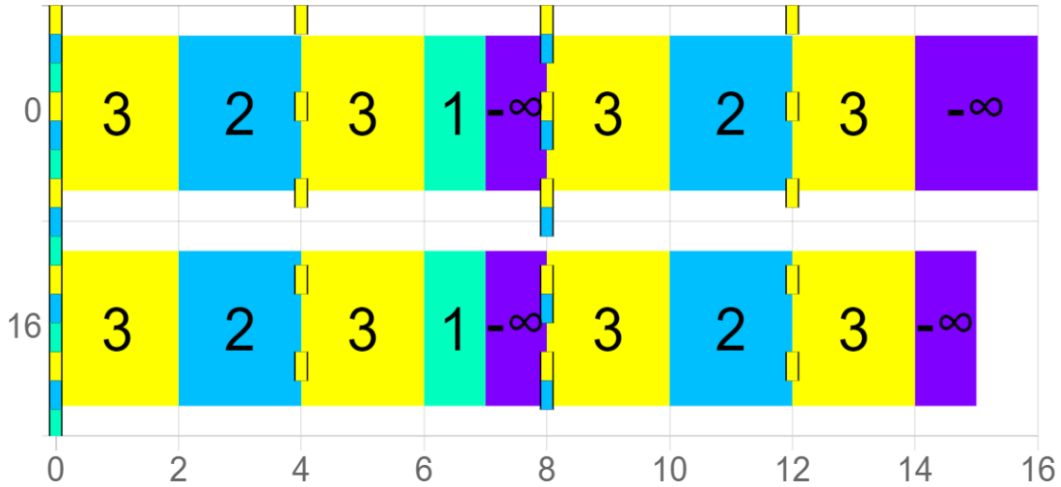
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	
2	0	4	4	3	0	
2	0	8	8	2	0	
1	0	16	16	1	0	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	

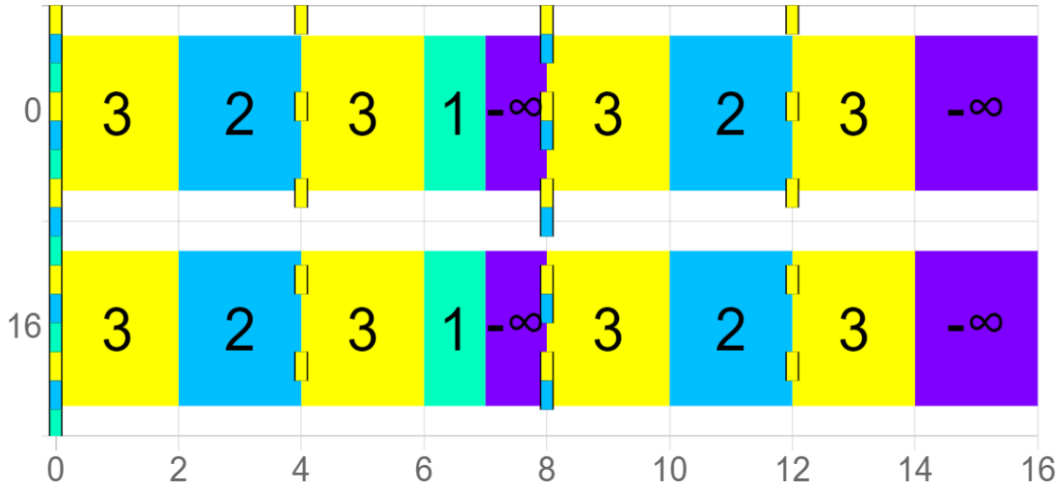
Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Fixed-Priority Preemptive Scheduling (FP)

- Ablaufplanung (vgl. SP2)
  - ☞ *“Wann darf welcher Prozess/Task laufen?”*
- Grundverfahren von Scheduling bei Echtzeitsystemen: FP



WCET	arrivalTime	deadline	period	priority	remainingWork	
2	0	4	4	3	0	
2	0	8	8	2	0	
1	0	16	16	1	0	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	

Task-Prioritäten:
$3 > 2 > 1 > -\infty$
$-\infty \triangleq$ Leerlauftask

Deadlines:
vertikale, gestrichelte Linien

https://www.simonlanger.com/uni/kvbk/

Grundlegende Begriffsklärungen und Motivation

Angriff: “ScheduLeak in Action”

Verteidigung: “TaskShuffler in Action”

Diskussion: Praktikabilität und Stabilität

Diskussion: Ausgelöste Probleme und Alternativen

Timing Inference: Angriff durch “ScheduLeak”

- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver

https://www.simonlanger.com/uni/kvBK/

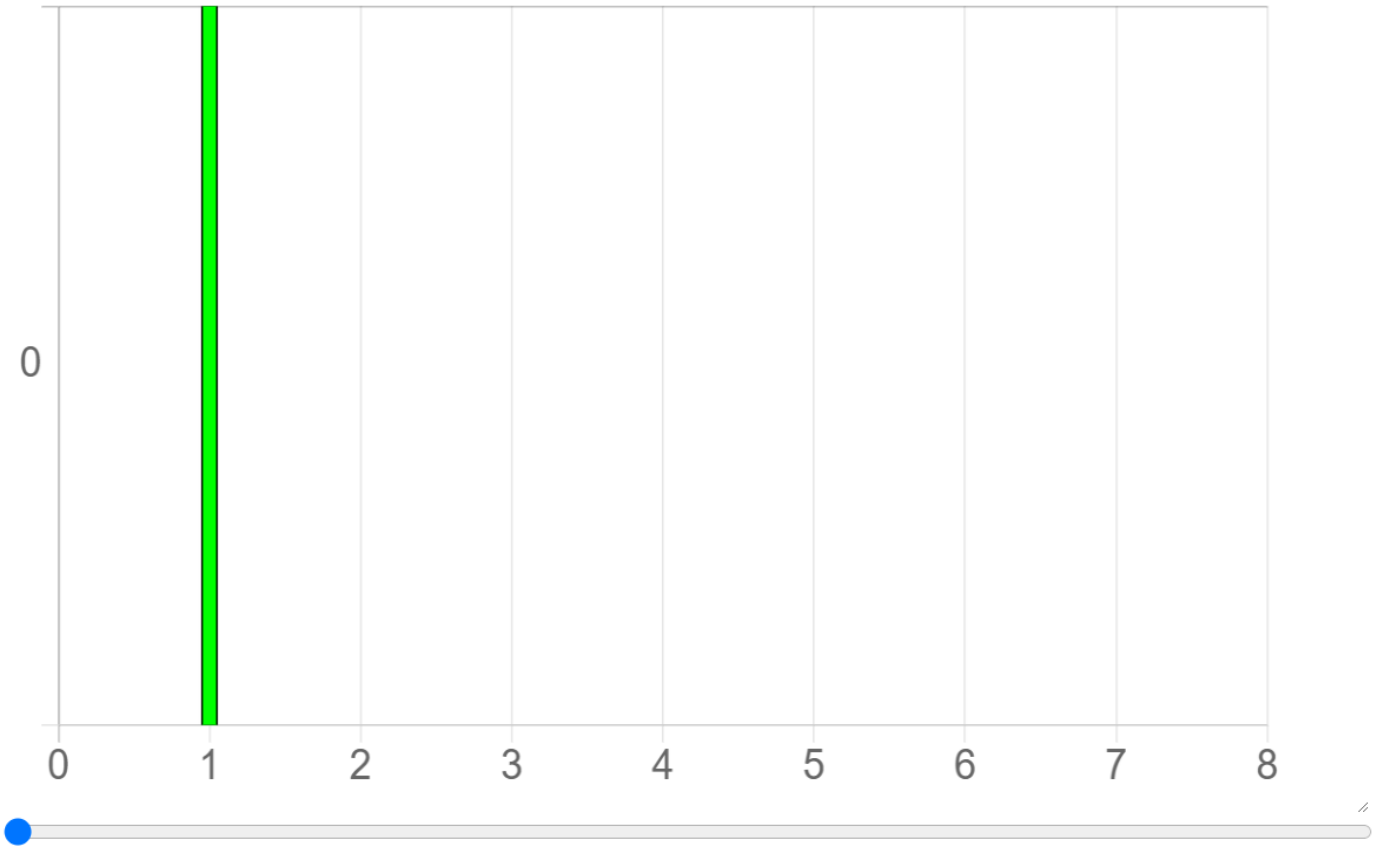
Timing Inference: Angriff durch “ScheduLeak”

- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver

https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

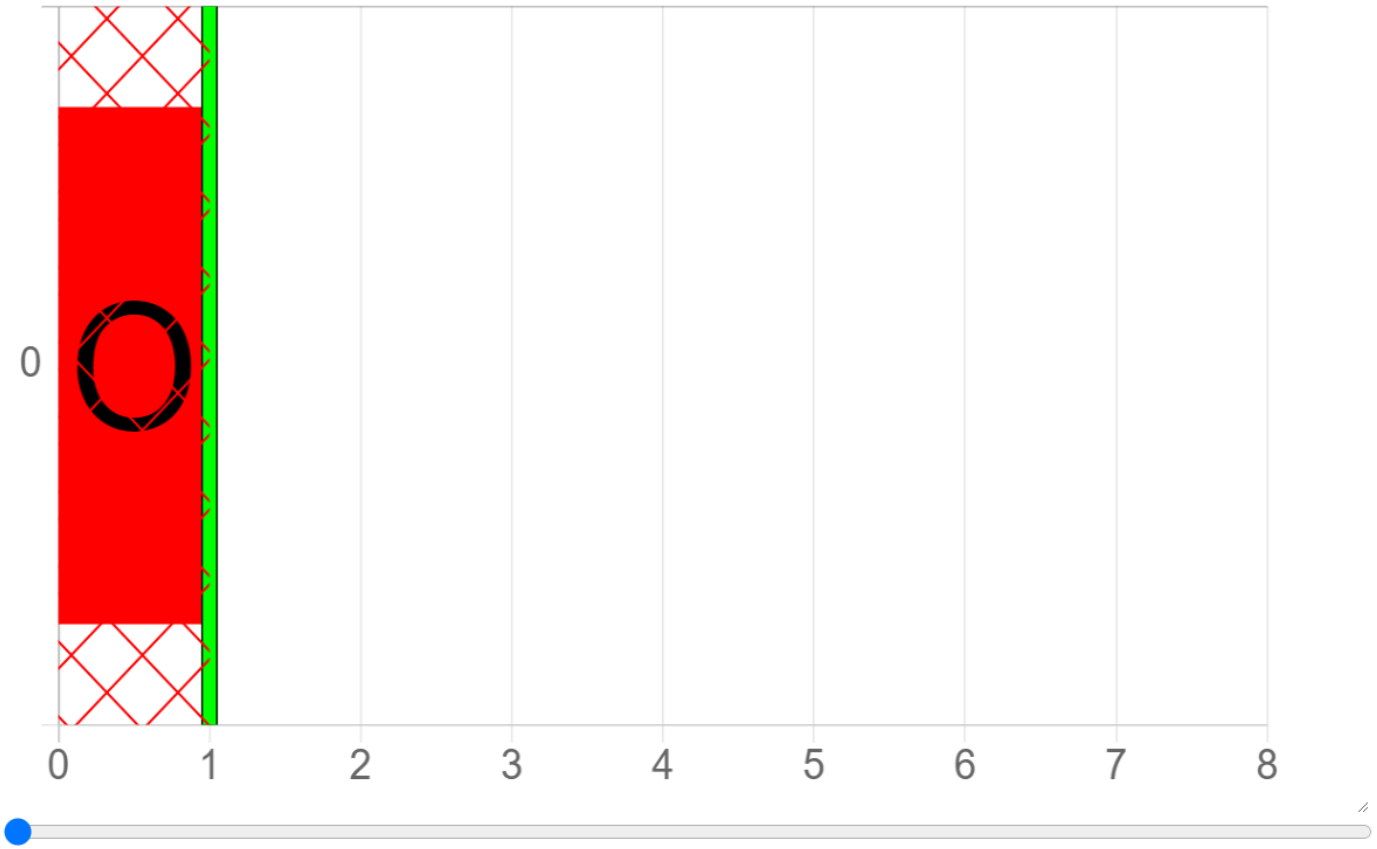
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

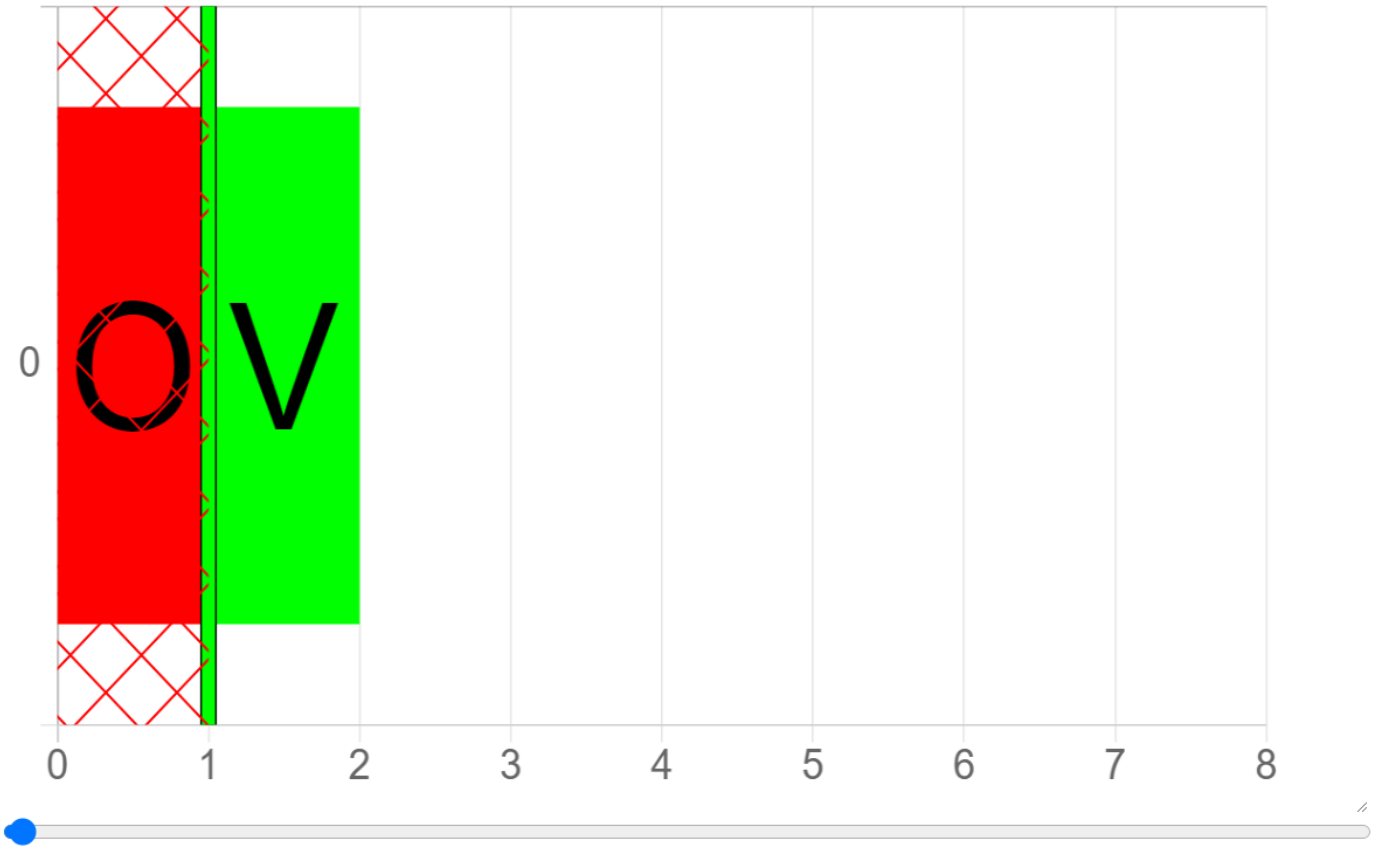
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

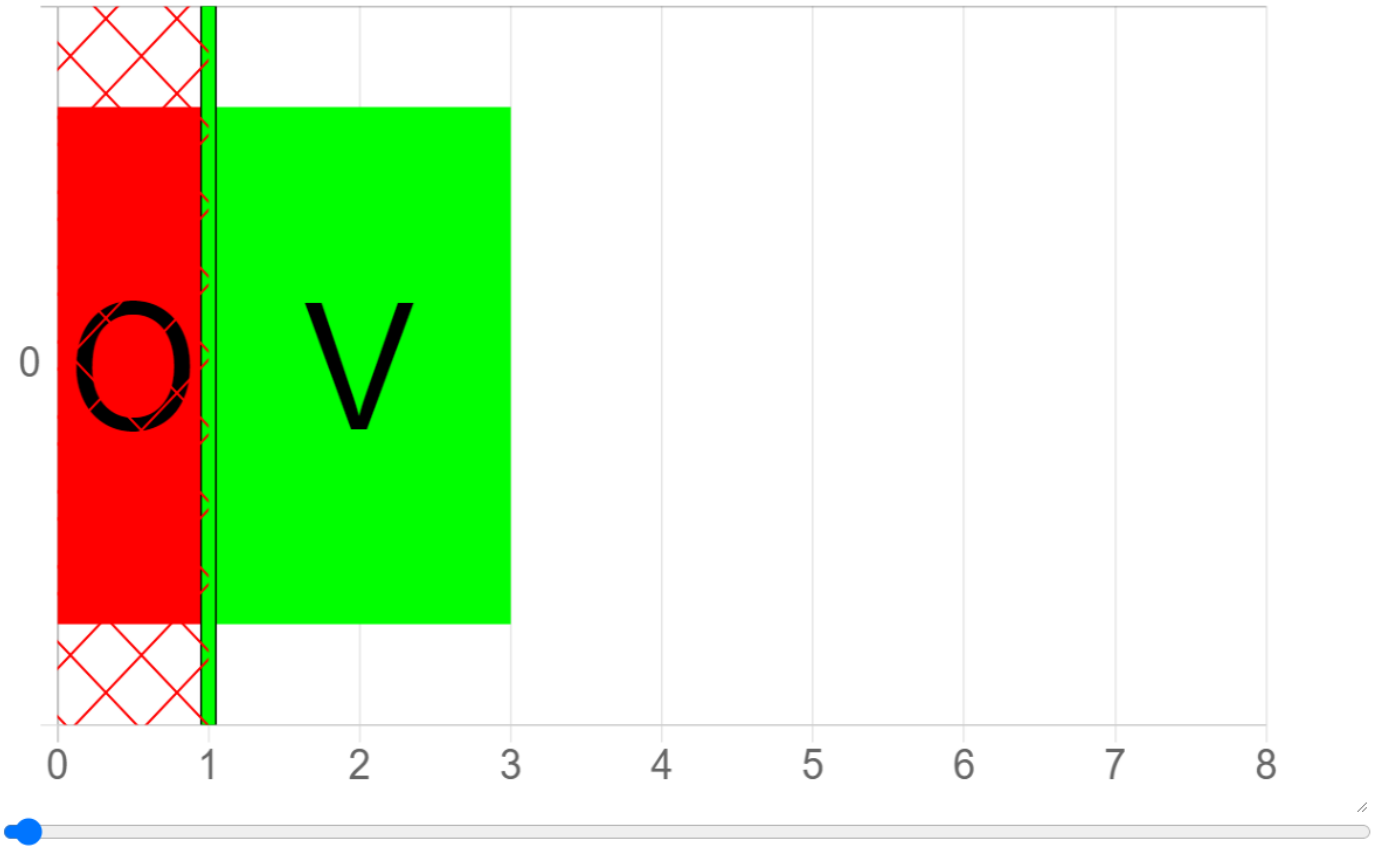
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

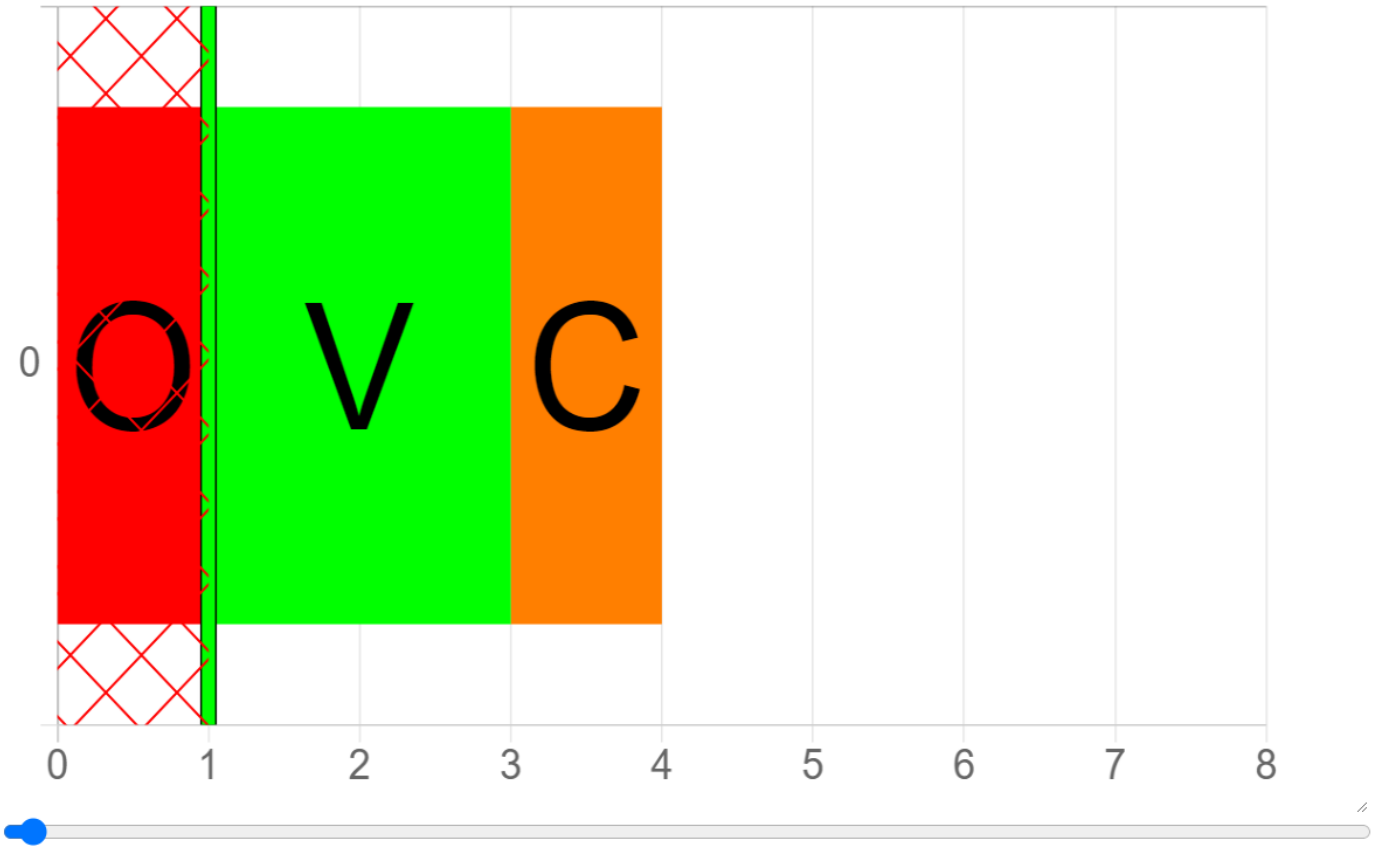
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

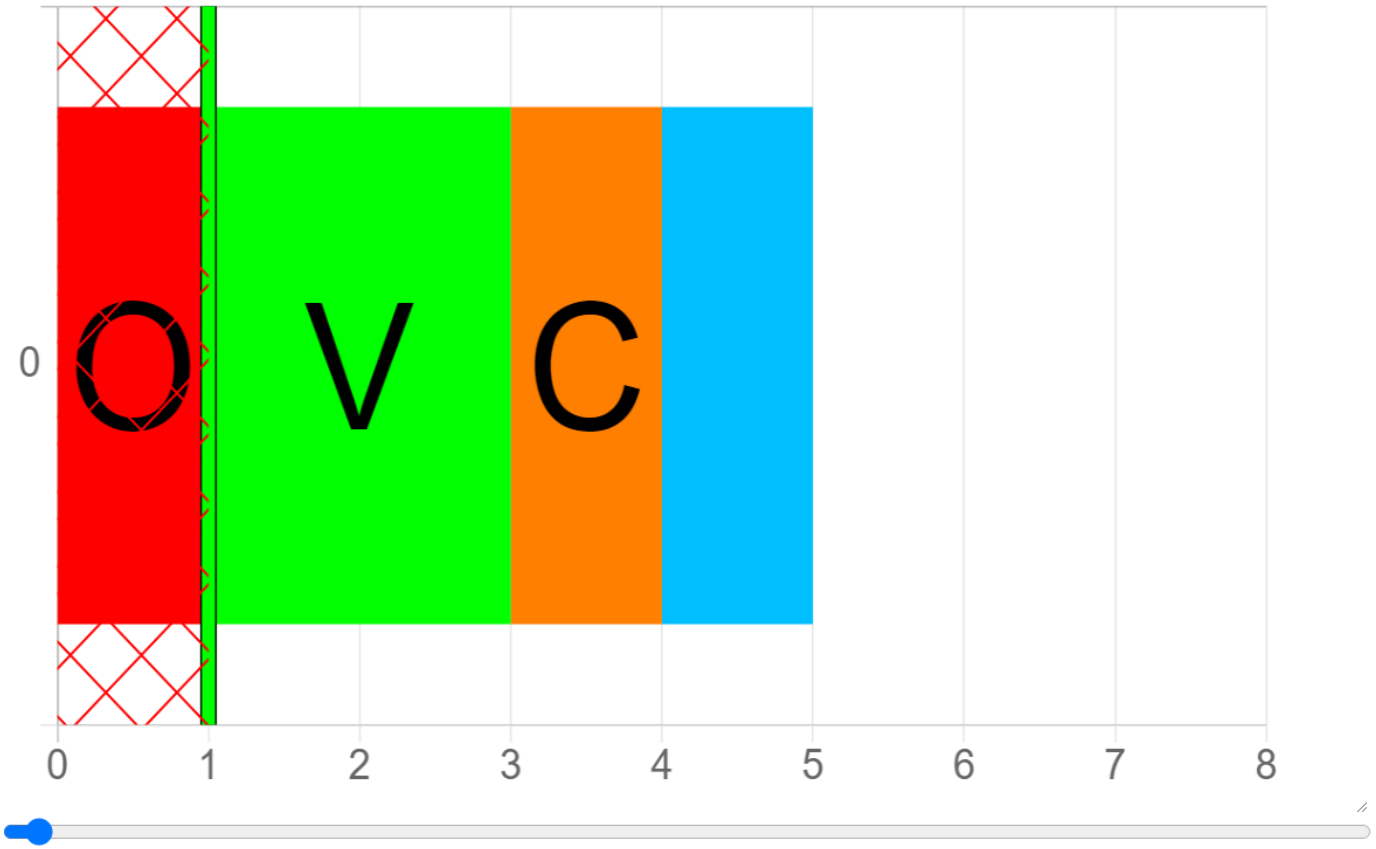
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

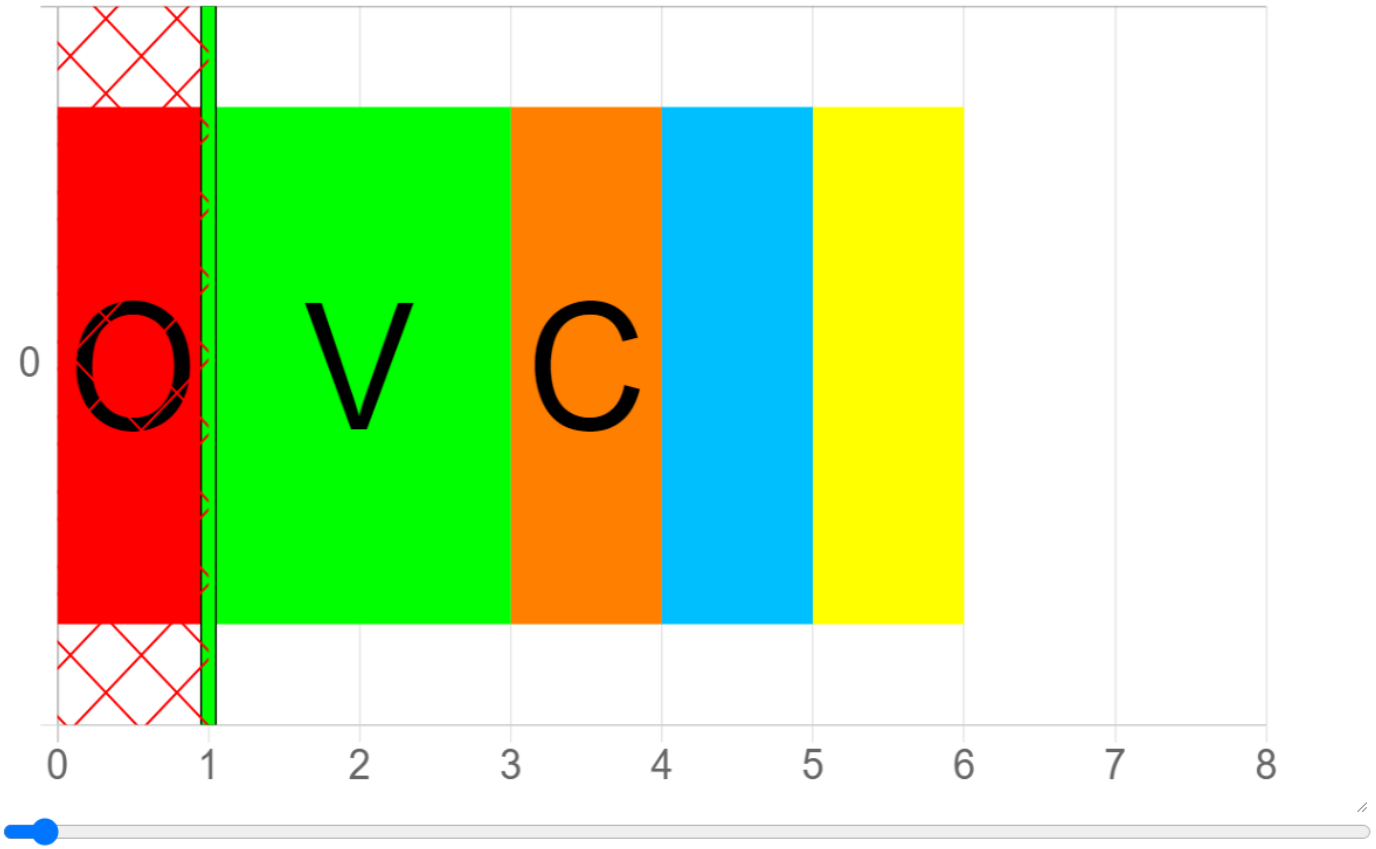
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

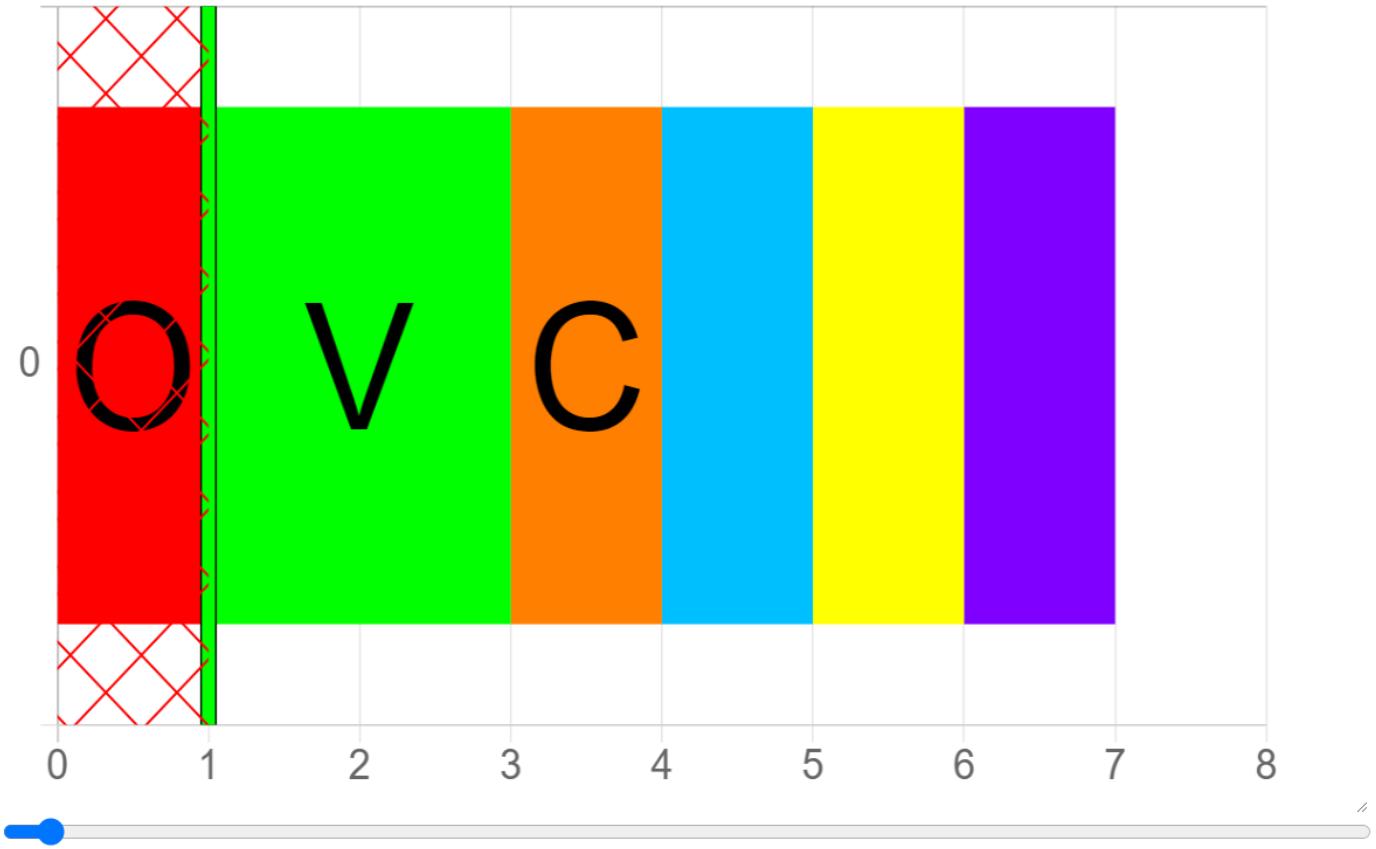
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

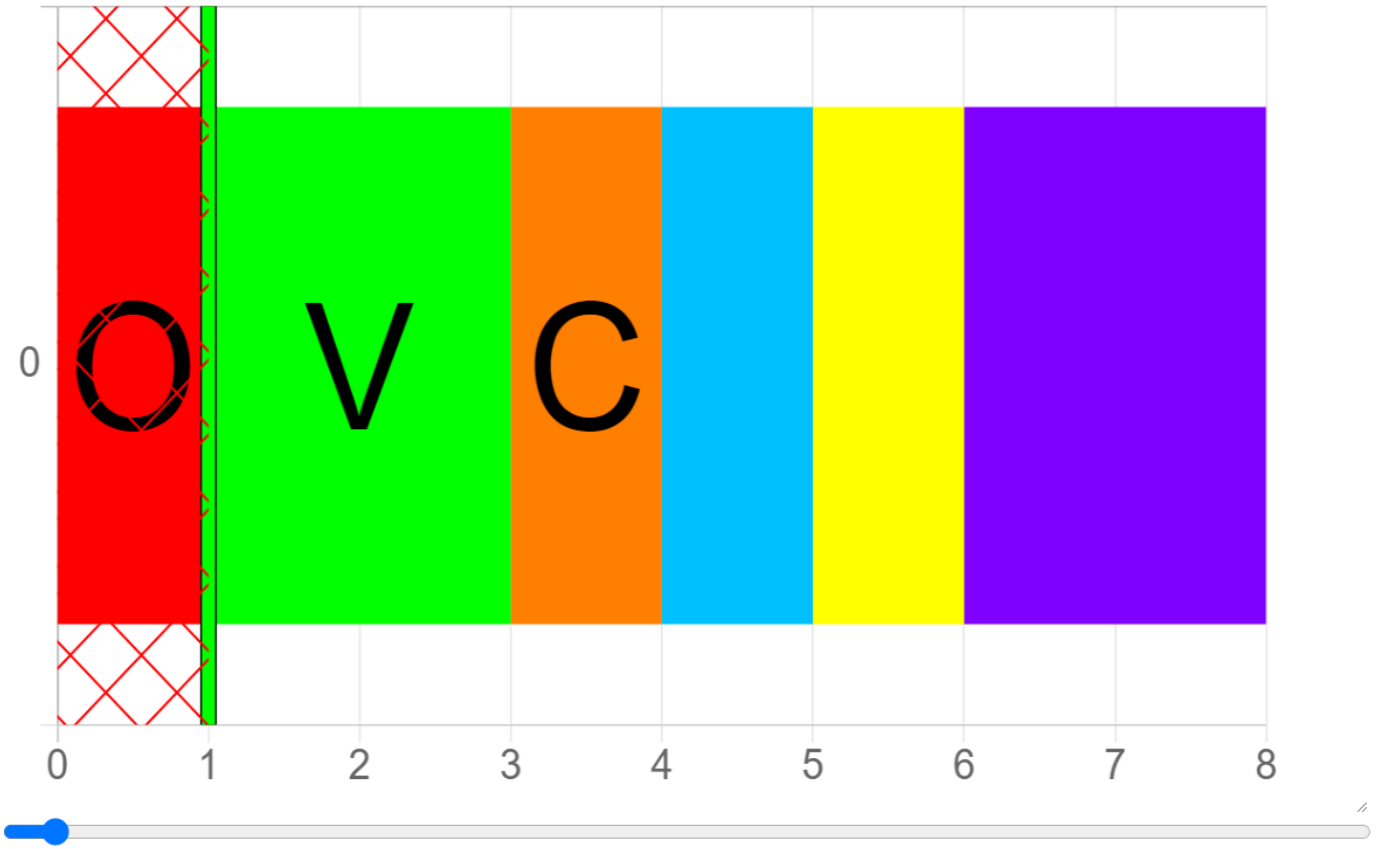
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

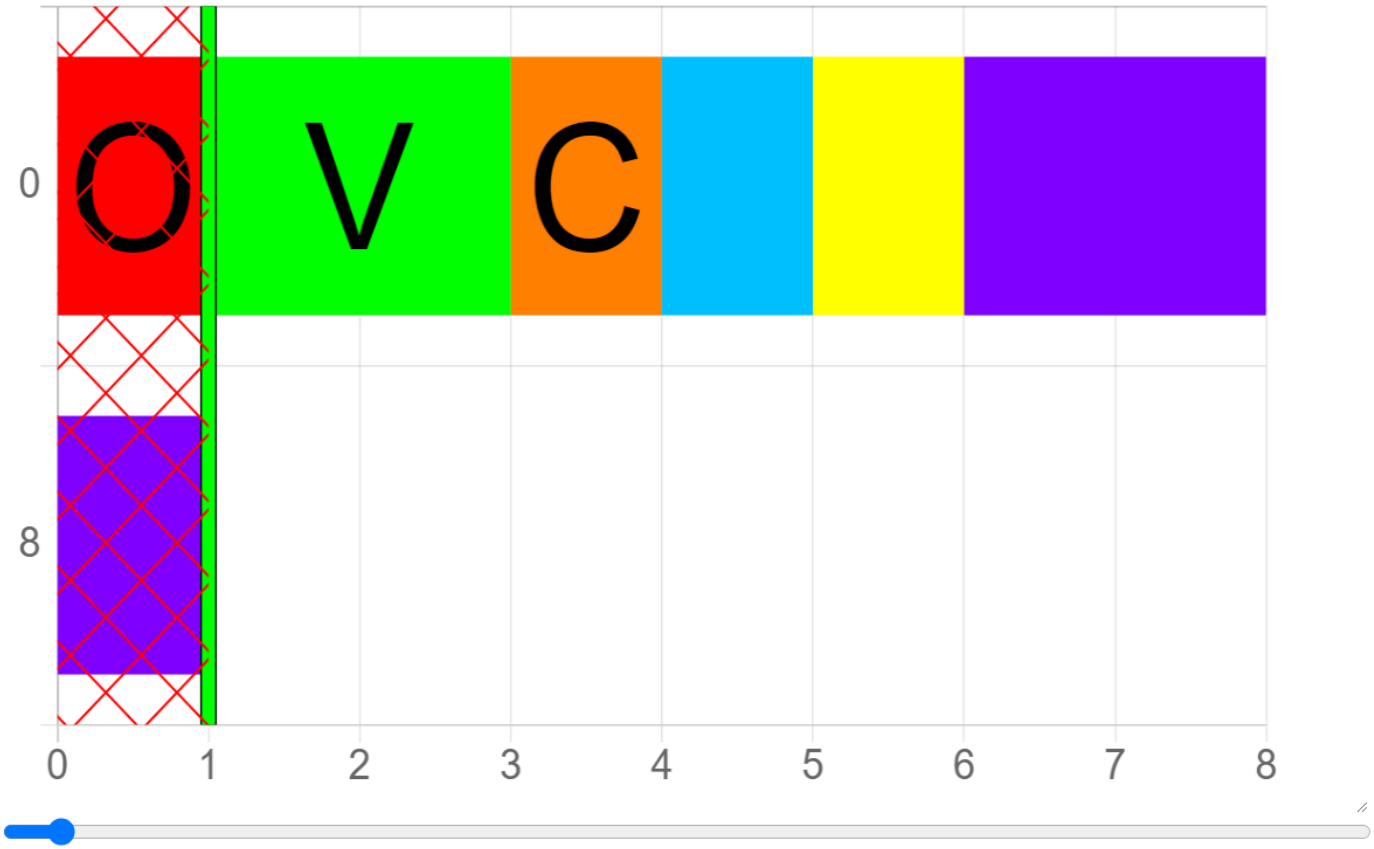
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

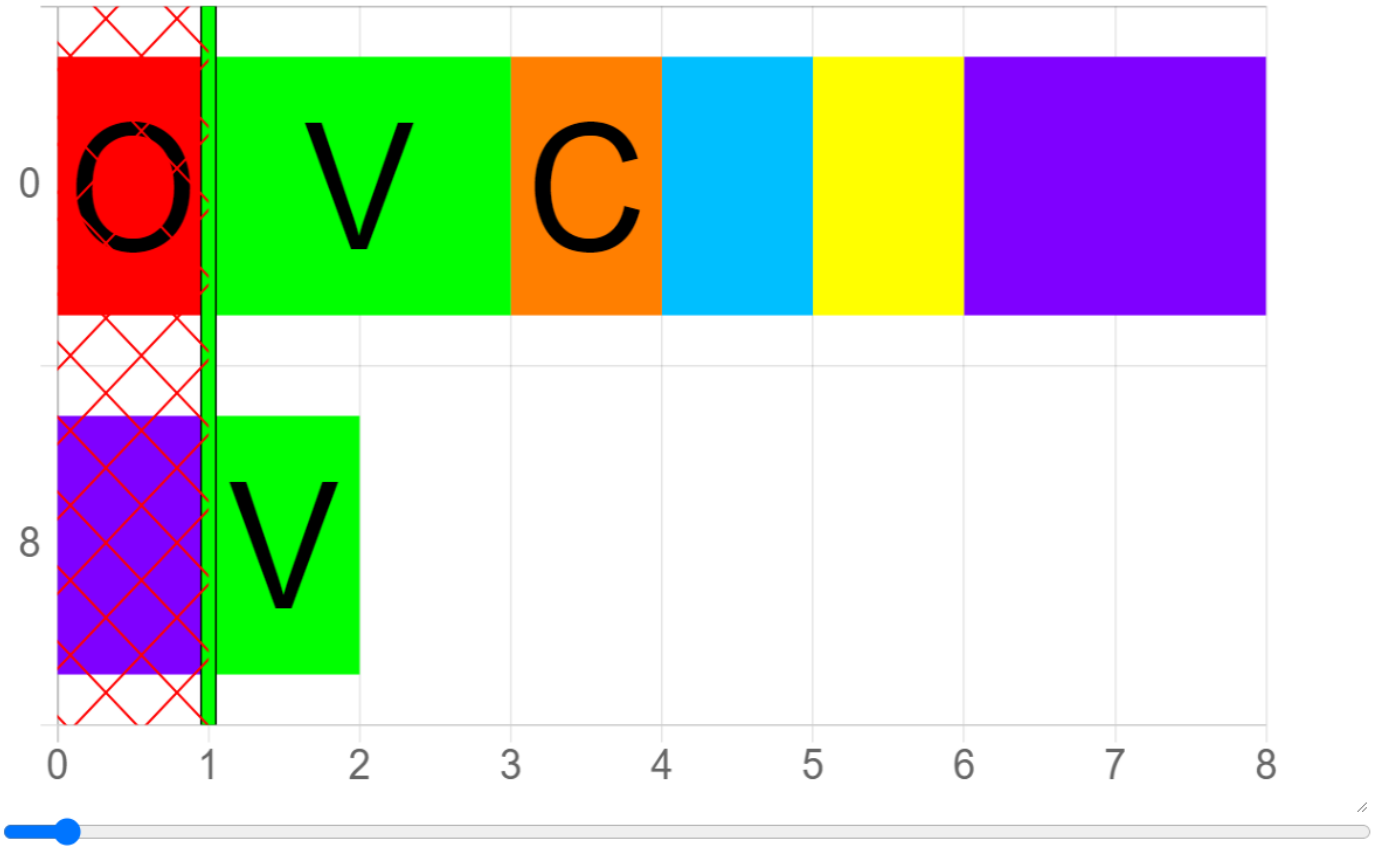
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

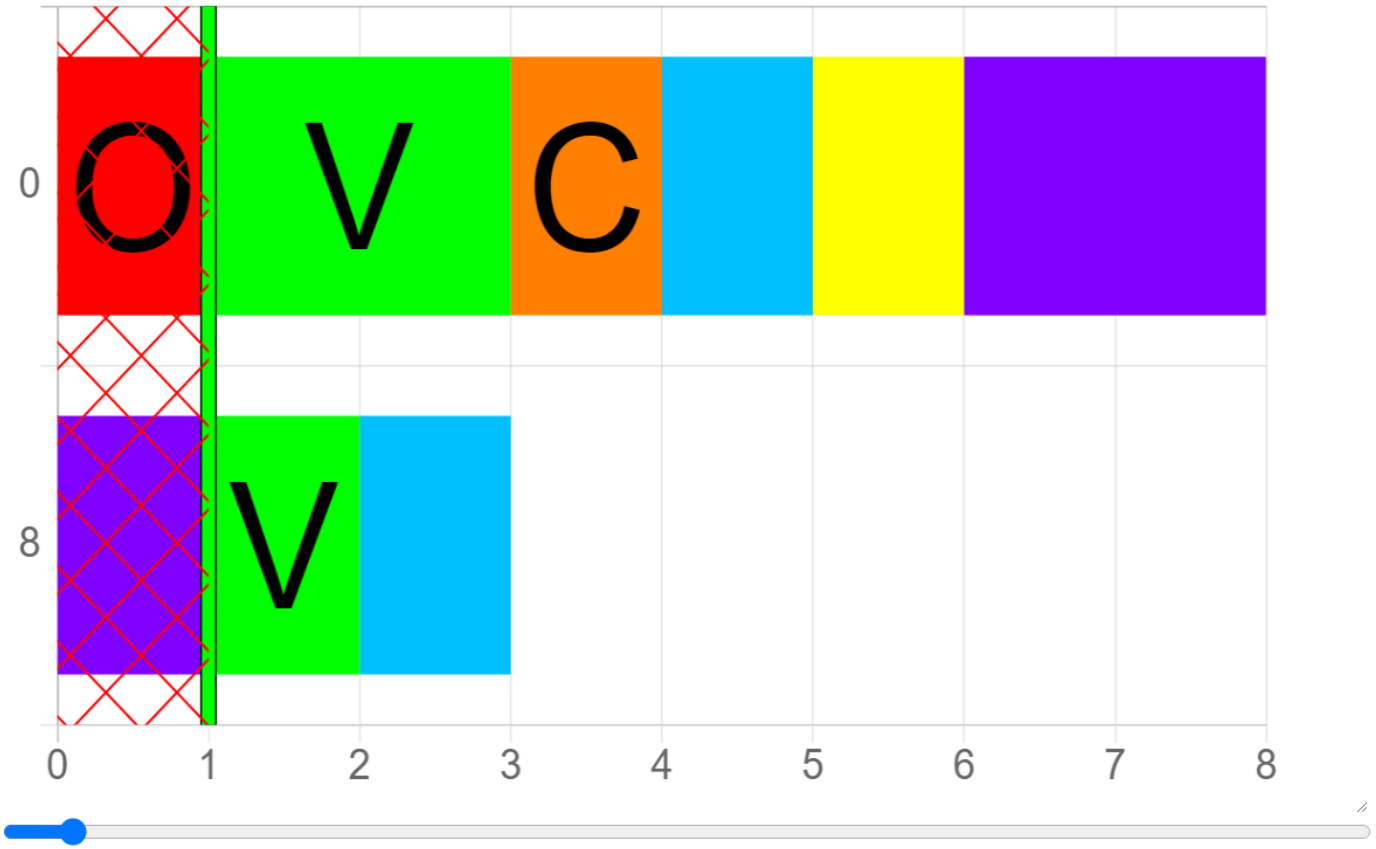
- Herleiten von (zeitlichen) Informationen durch Angreifer = Seitenkanalangriff (**Side-Channel Attack**)
  - ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

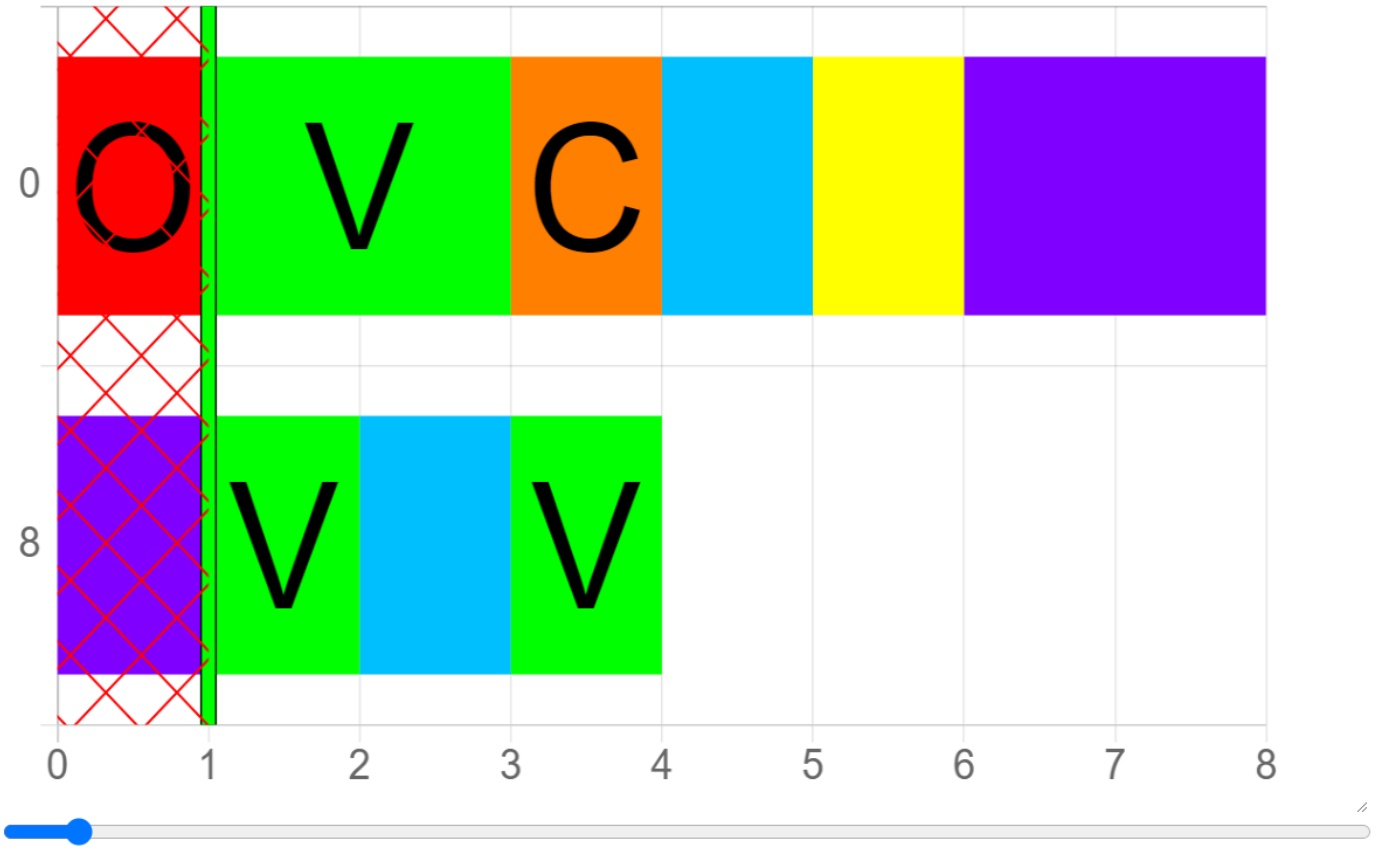
- Herleiten von (zeitlichen) Informationen durch Angreifer = Seitenkanalangriff (**Side-Channel Attack**)
  - ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

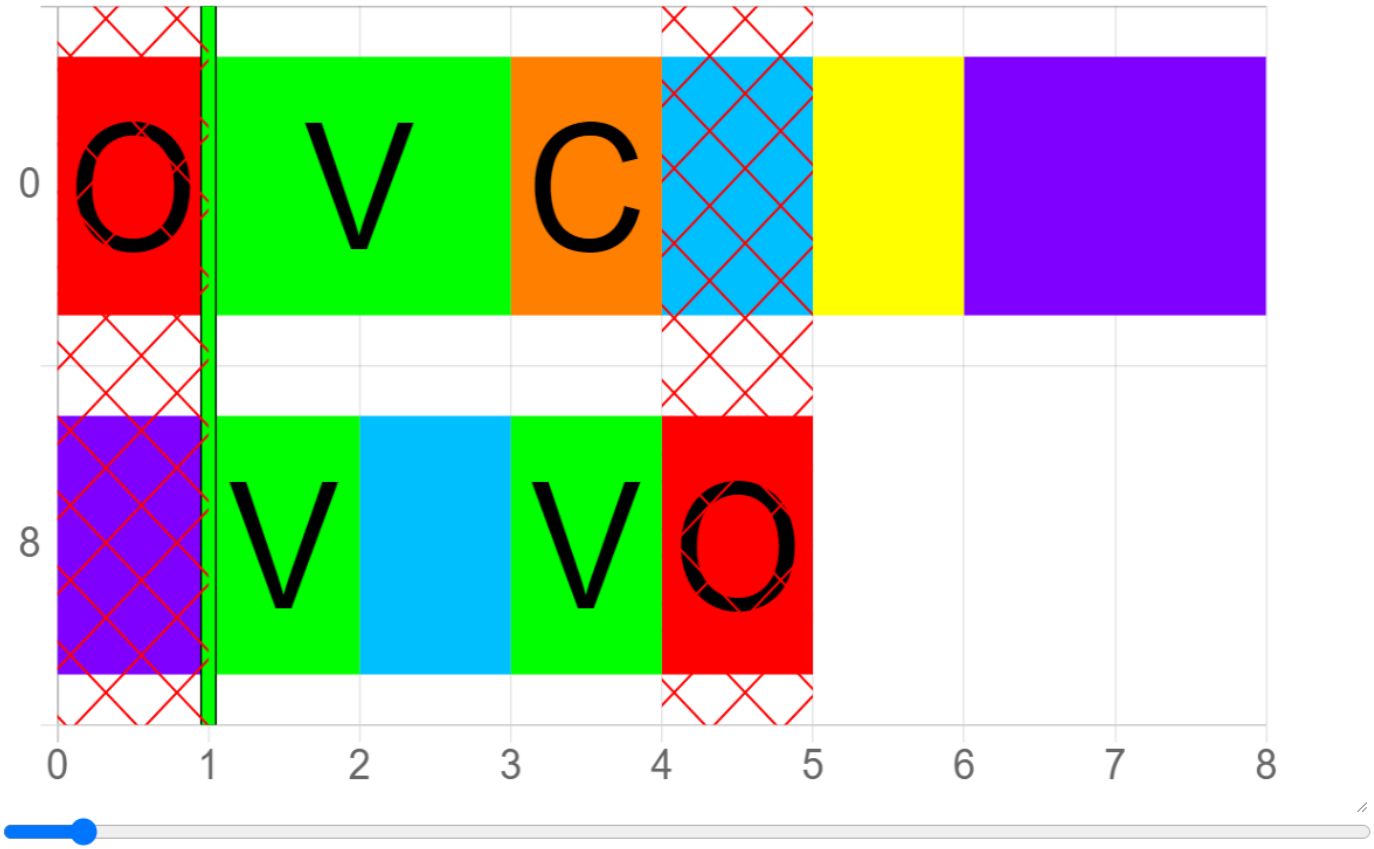
- Herleiten von (zeitlichen) Informationen durch Angreifer = Seitenkanalangriff (**Side-Channel Attack**)
  - ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

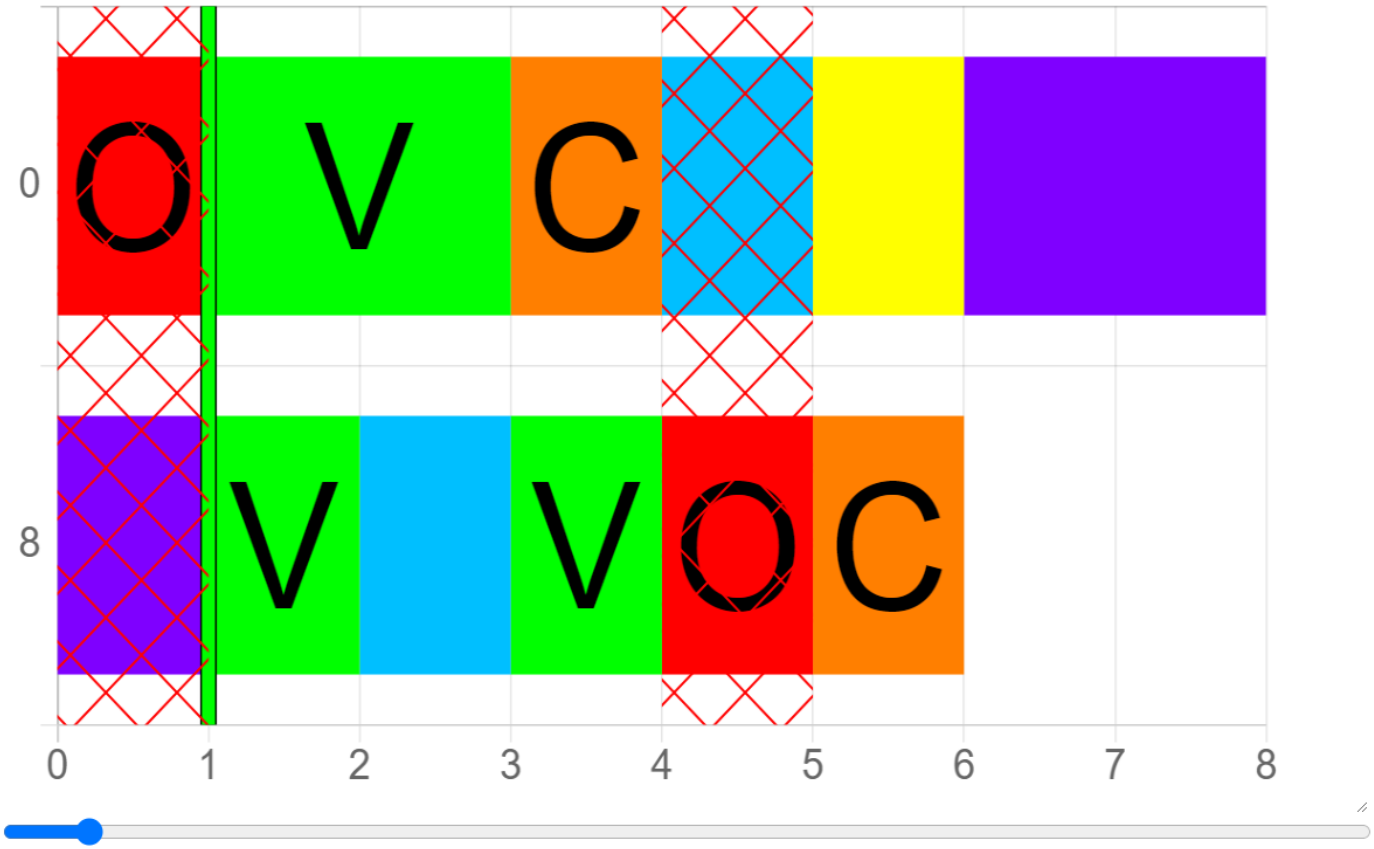
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

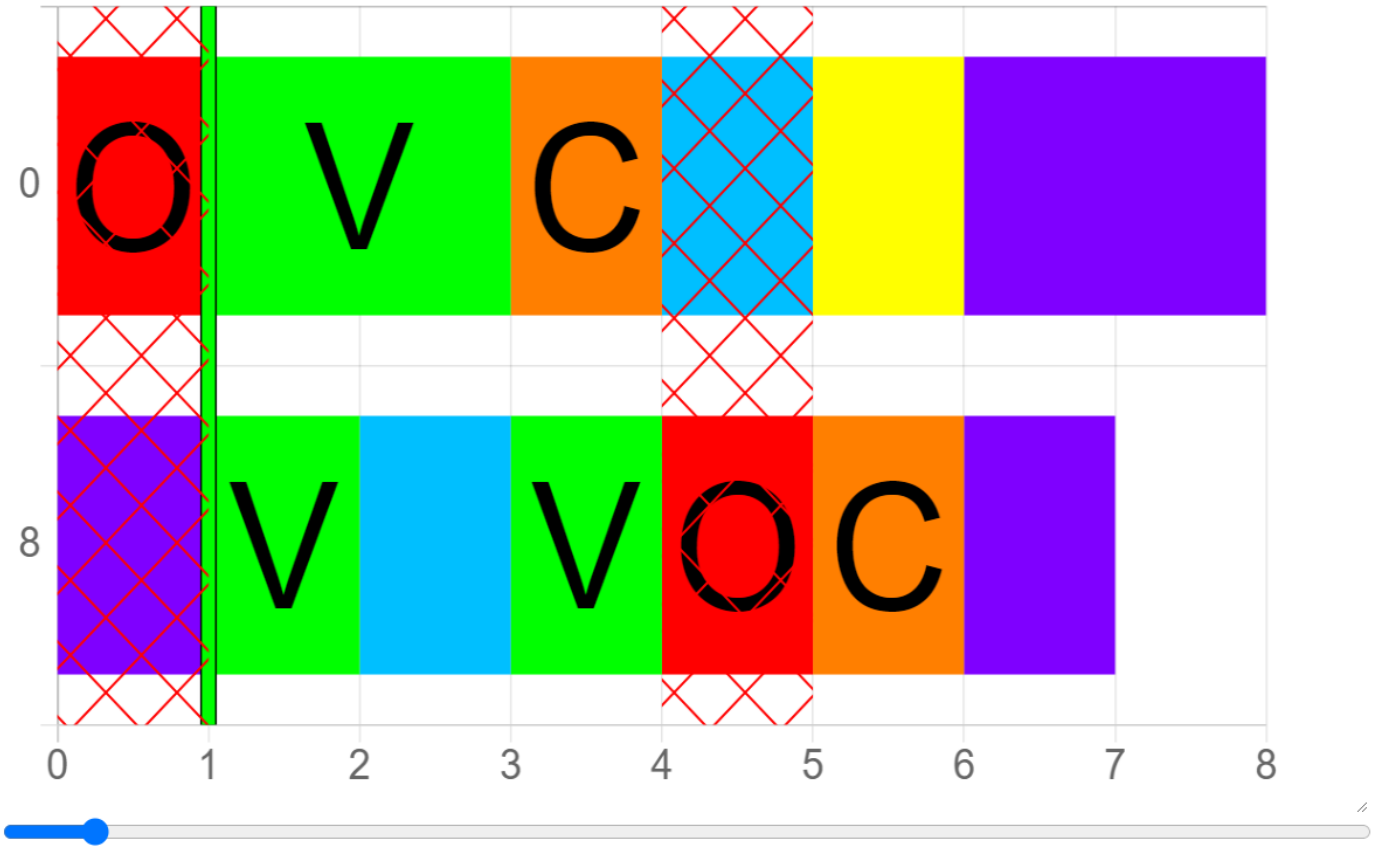
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

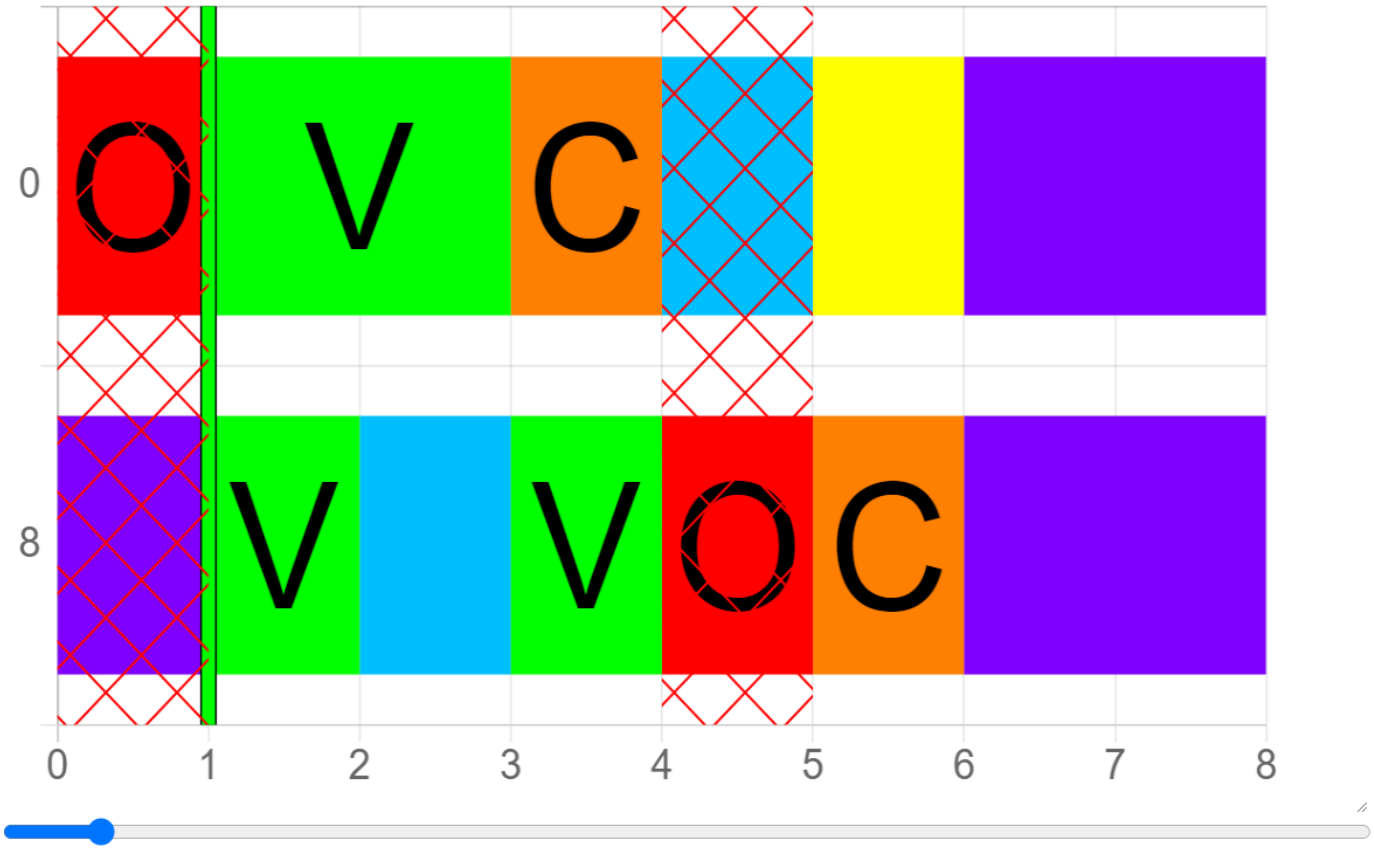
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

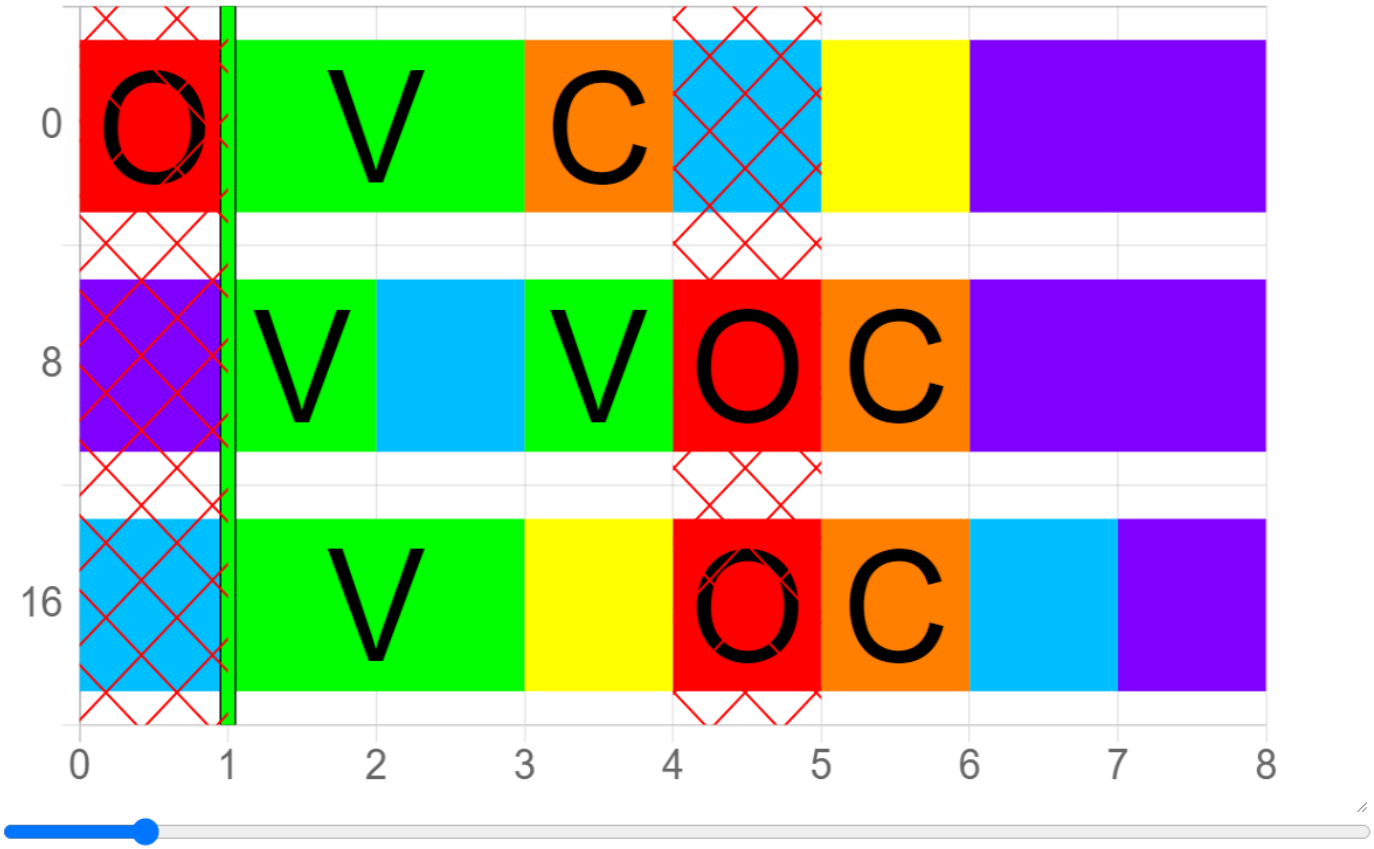
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

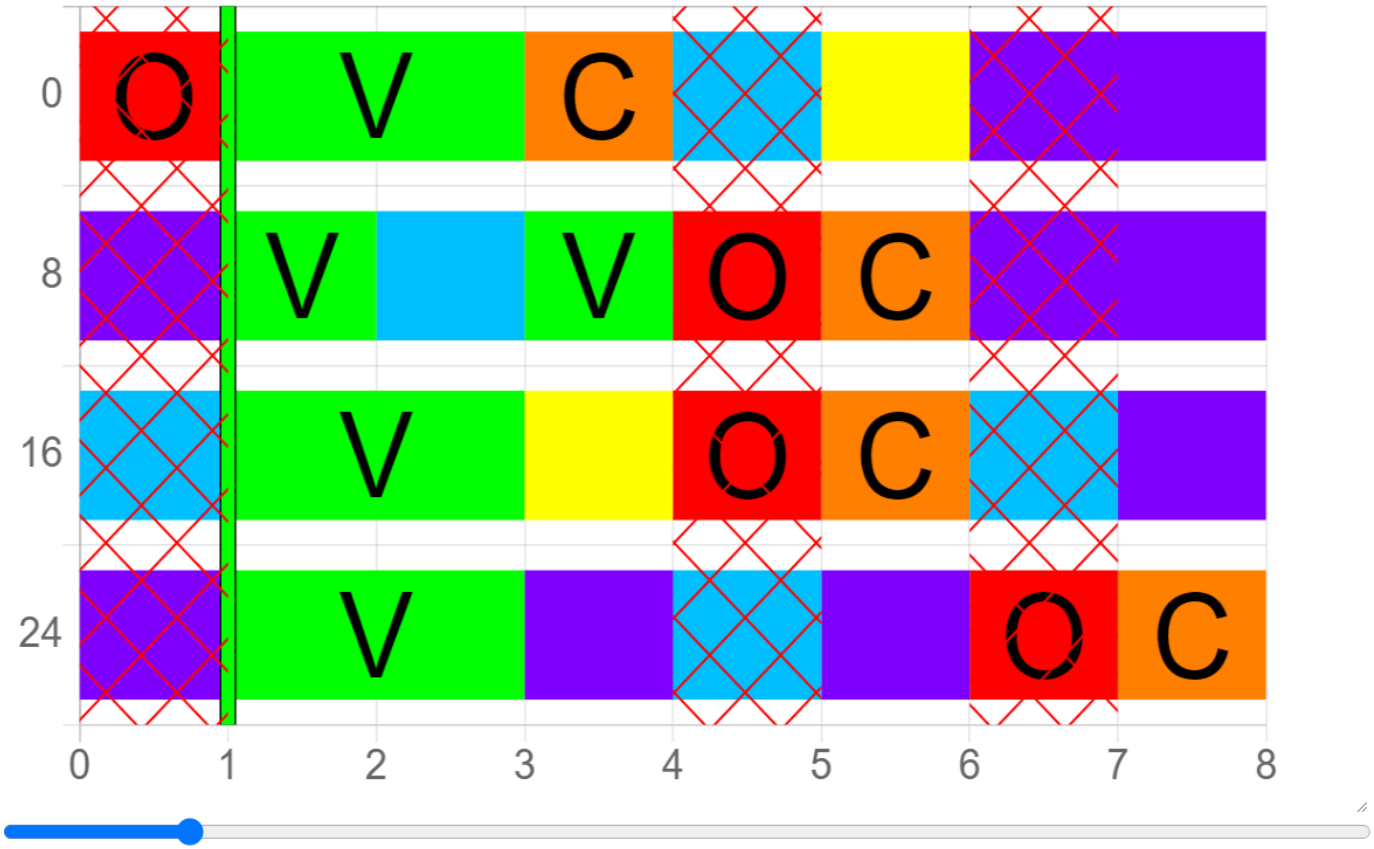
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

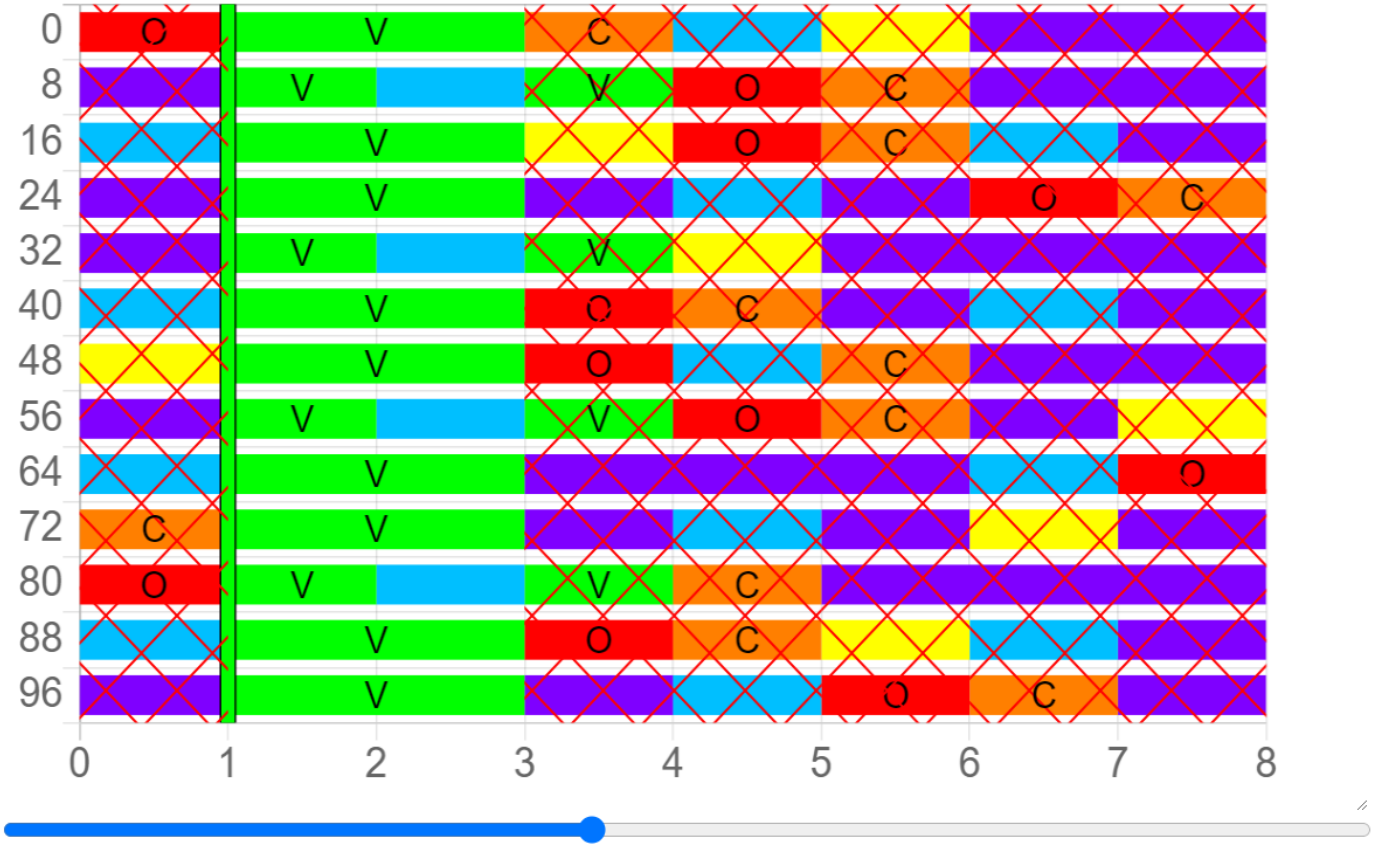
- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



https://www.simonlanger.com/uni/kvBK/

Timing Inference: Angriff durch “ScheduLeak”

- Herleiten von (zeitlichen) Informationen durch Angreifer
  = Seitenkanalangriff (**Side-Channel Attack**)
  ☞ *“Wann wird mein Angriffsziel laufen?”*
- (realist.) Prämisse: Priorität **V**ictim > Priorität **O**bserver



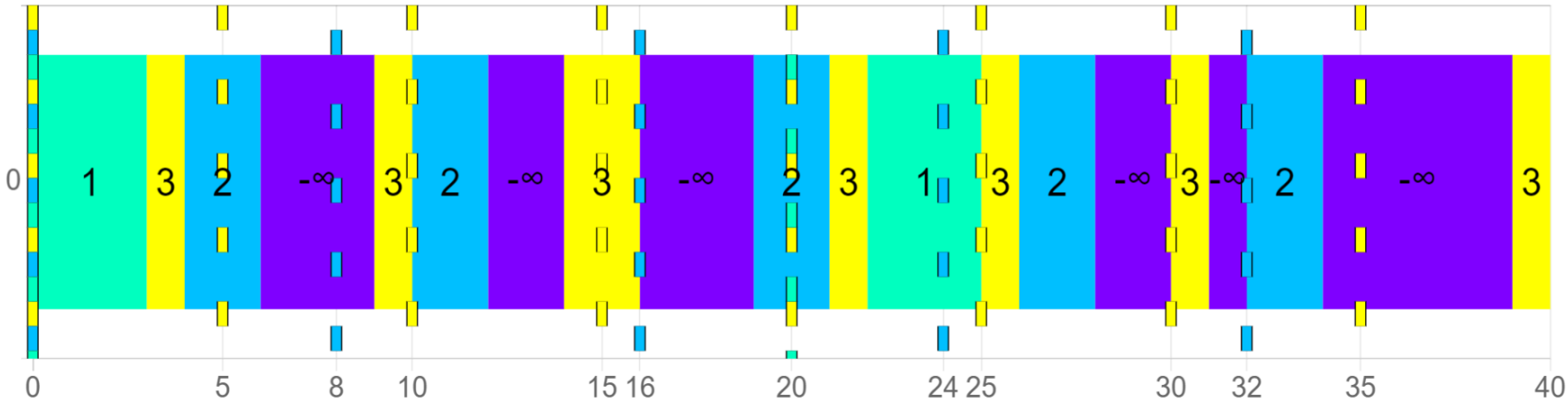
https://www.simonlanger.com/uni/kvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

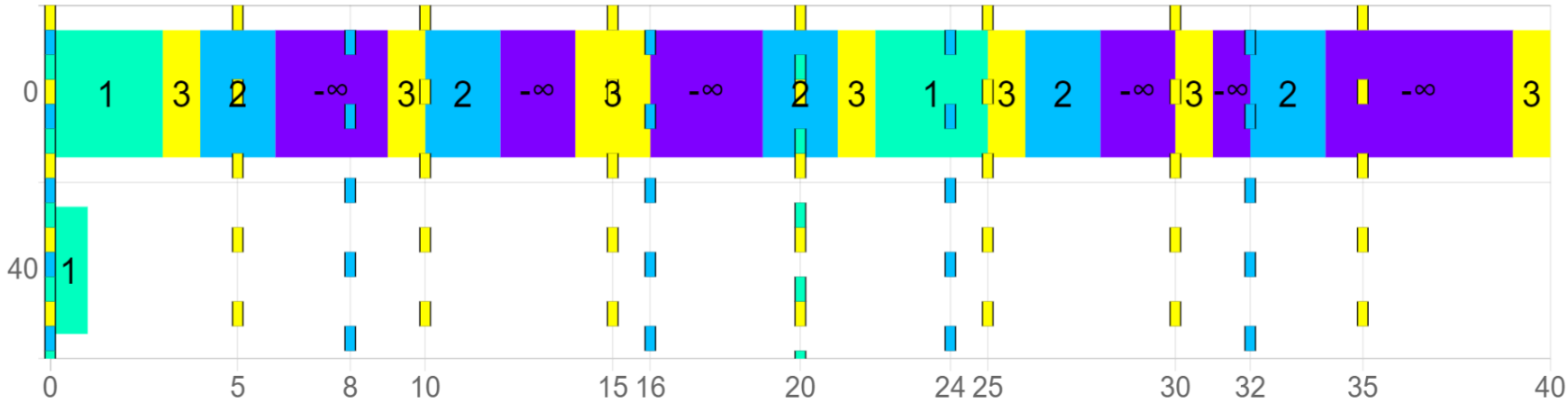


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	0	-Infinity	−
2	0	8	8	2	0	3	3	-Infinity	−
3	0	20	20	1	0	4	4	-Infinity	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	−

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

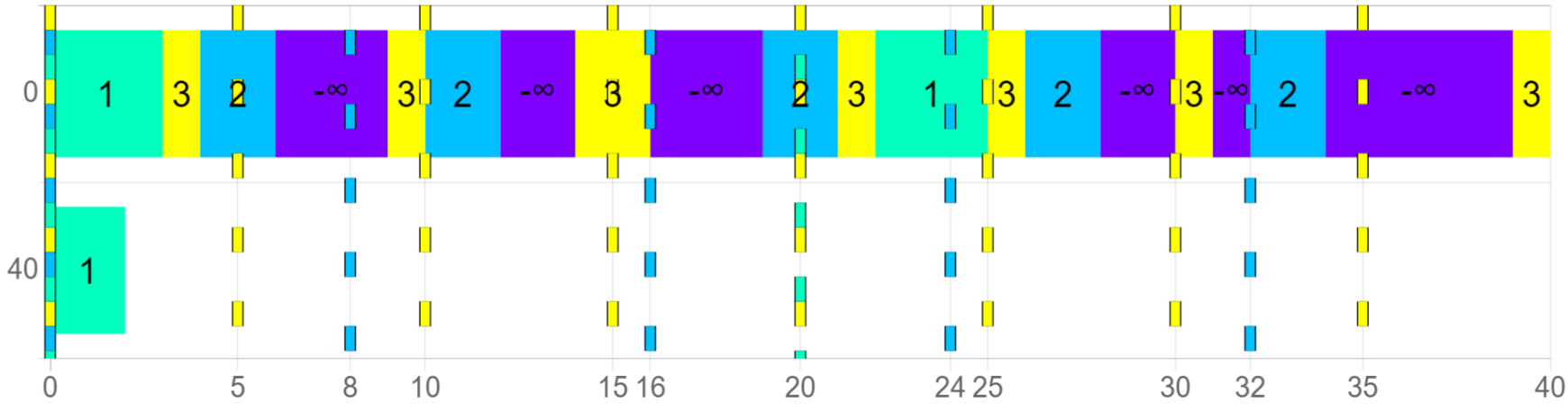


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	3	-Infinity	
2	0	8	8	2	2	3	2	-Infinity	
3	0	20	20	1	2	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

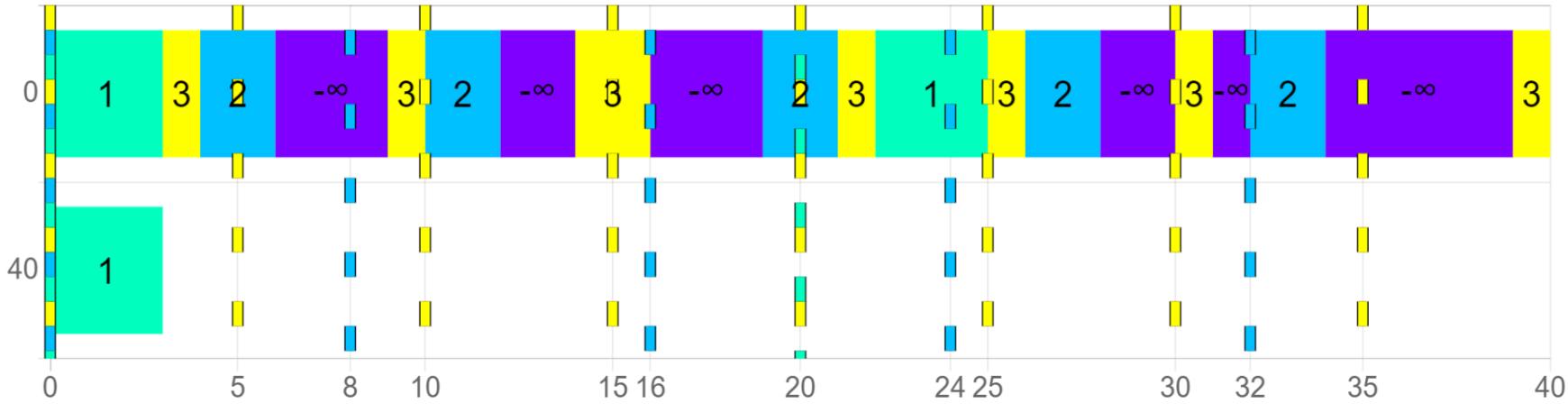


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	2	-Infinity	
2	0	8	8	2	2	3	1	-Infinity	
3	0	20	20	1	1	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

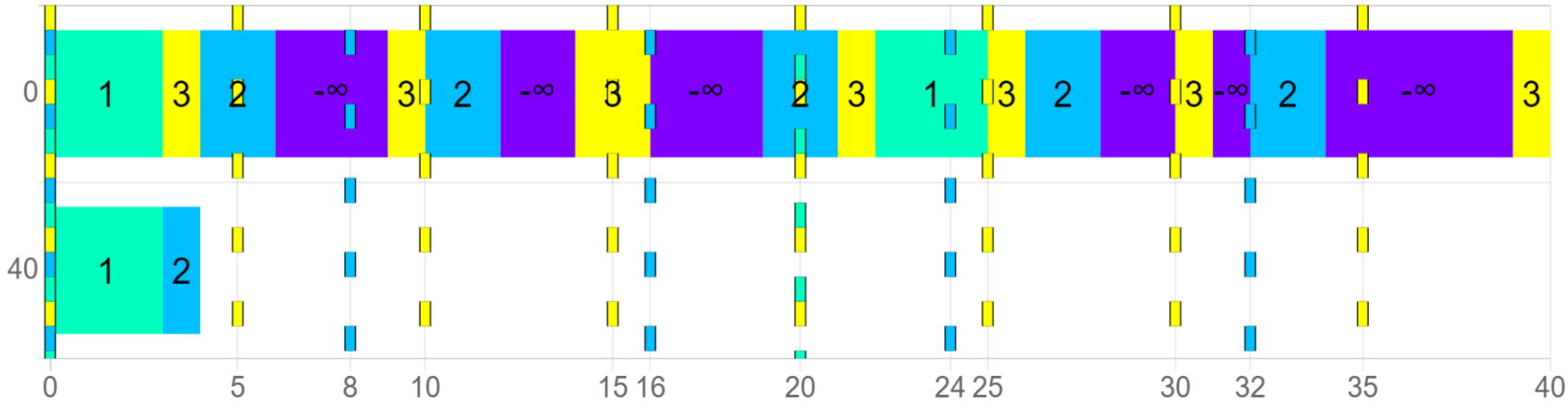


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	1	-Infinity	
2	0	8	8	2	2	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

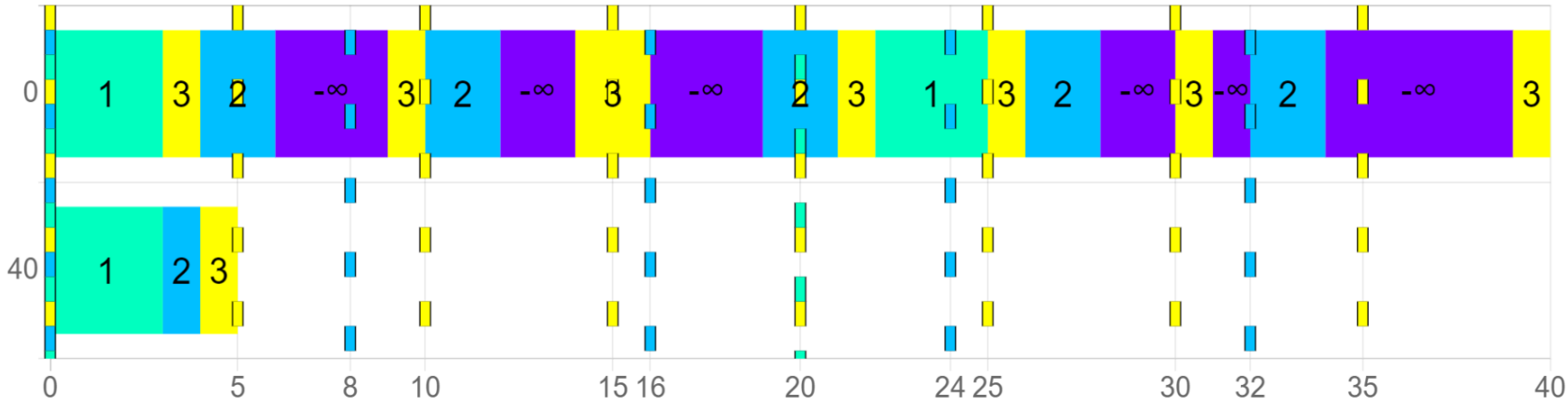


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	0	-Infinity	
2	0	8	8	2	1	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

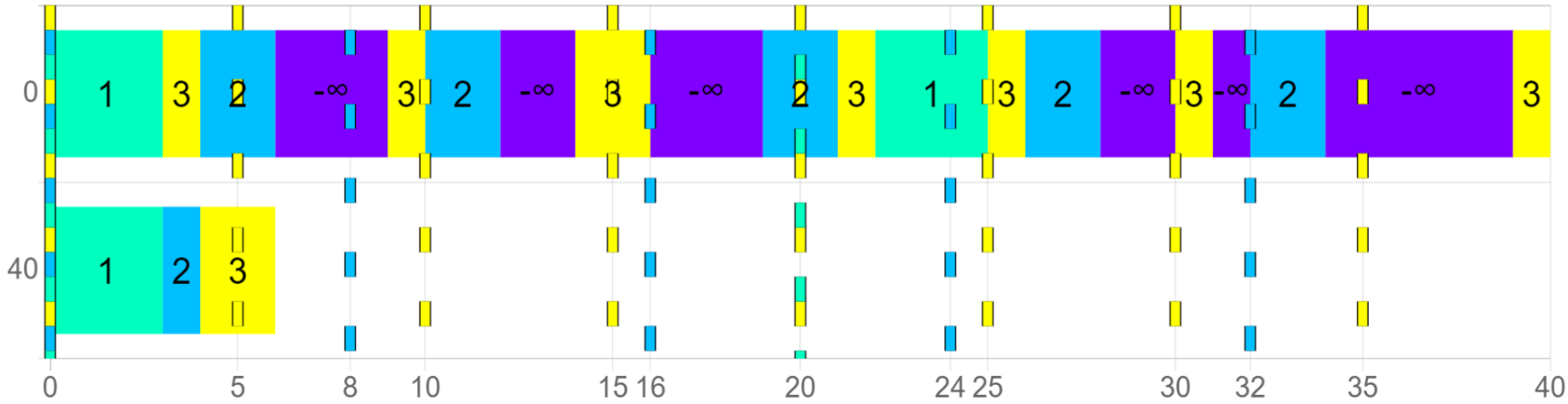


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	0	-Infinity	−
2	0	8	8	2	1	3	0	-Infinity	−
3	0	20	20	1	0	4	4	-Infinity	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	−

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

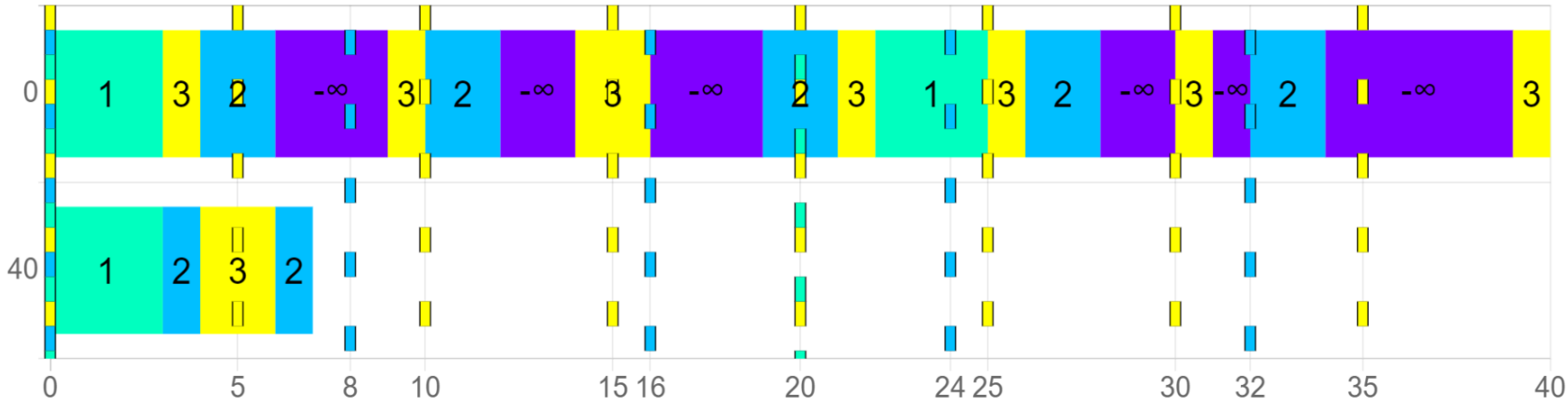


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	4	-Infinity	
2	0	8	8	2	1	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

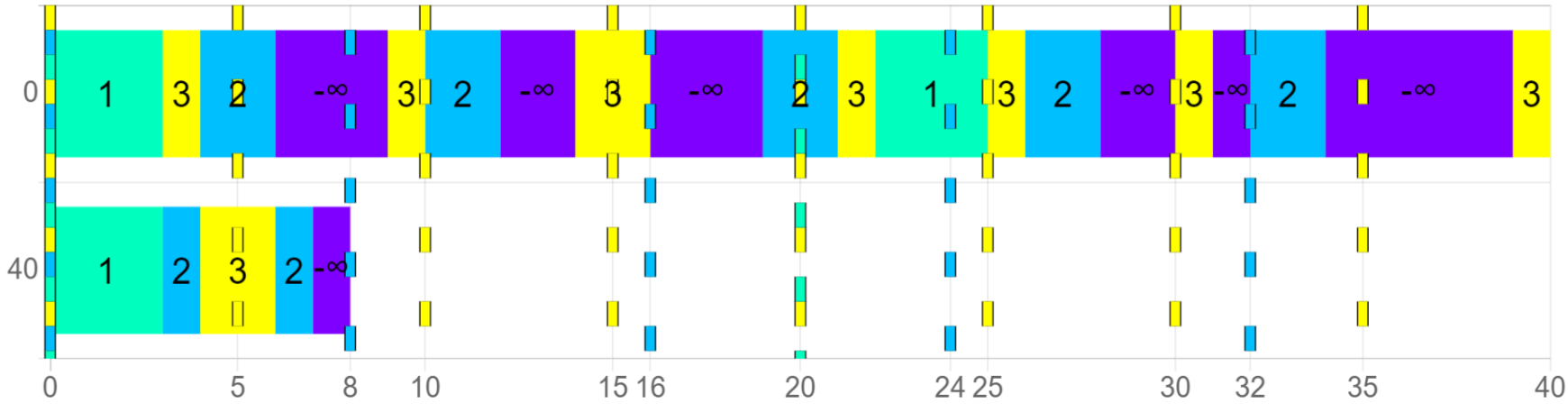


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	4	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

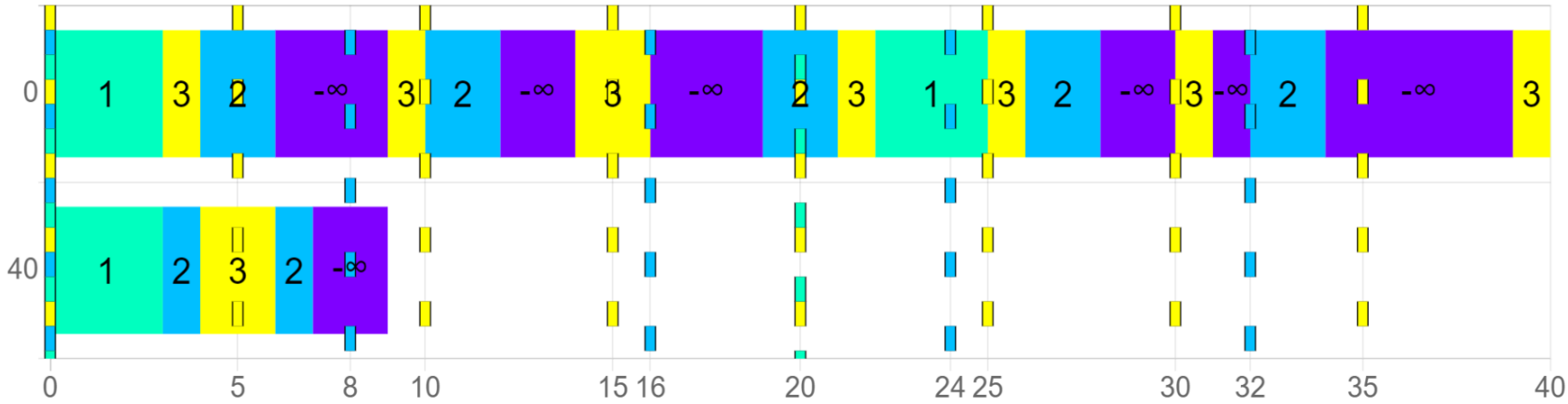


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	4	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

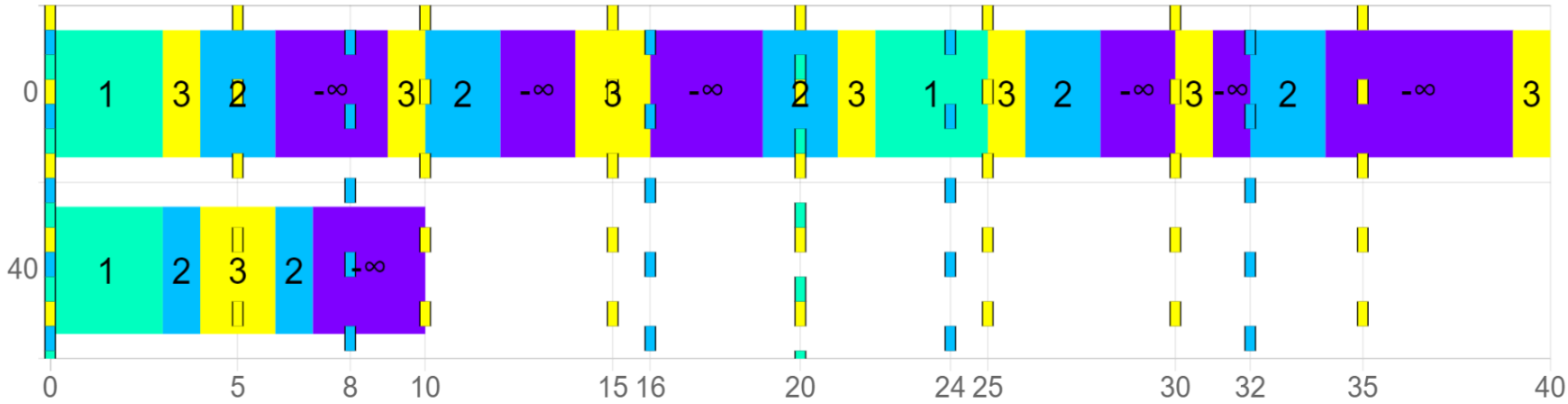


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio
1	0	5	5	3	0	4	4	-Infinity
2	0	8	8	2	2	3	2	-Infinity
3	0	20	20	1	0	4	4	-Infinity
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

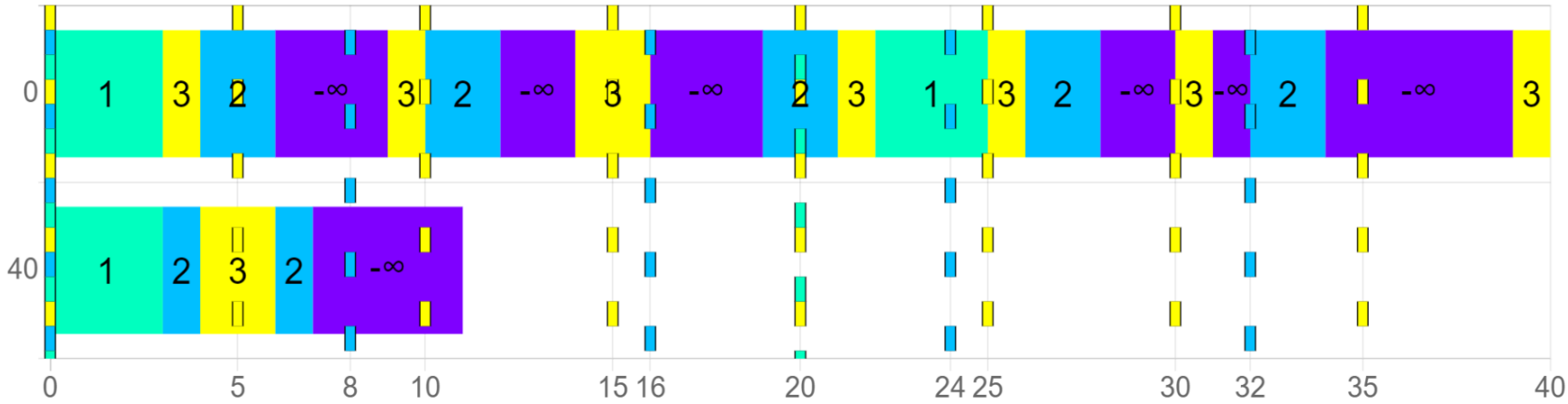


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	4	-Infinity	
2	0	8	8	2	2	3	1	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

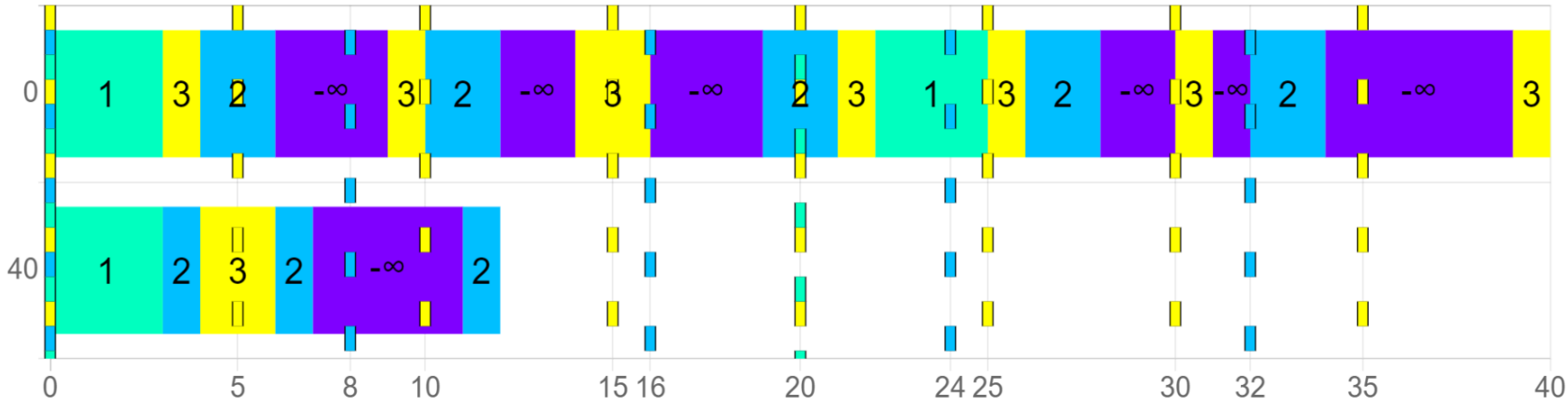


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	3	-Infinity	
2	0	8	8	2	2	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

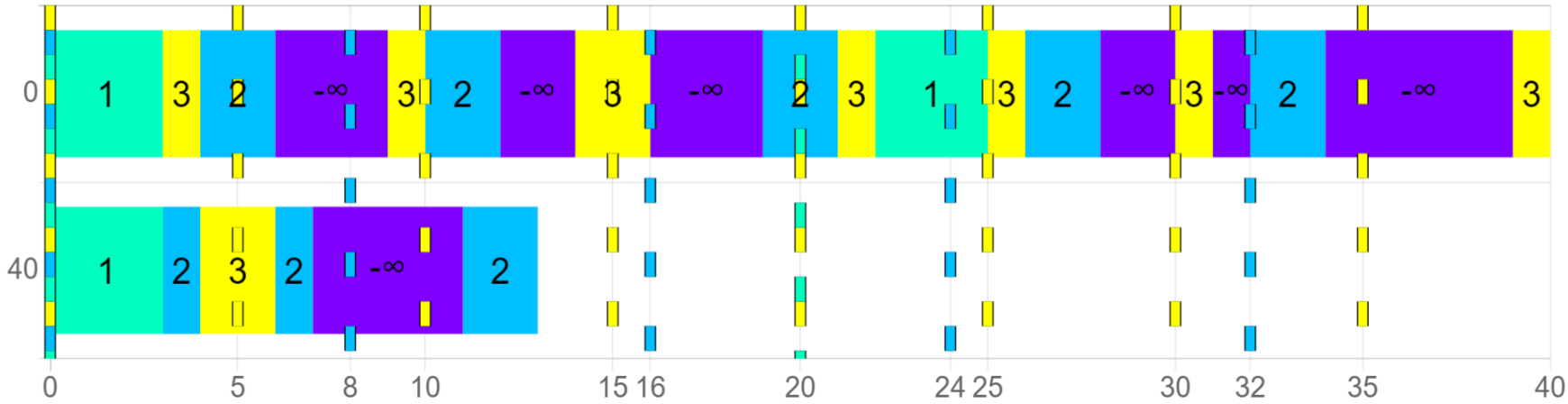


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	2	-Infinity	
2	0	8	8	2	1	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

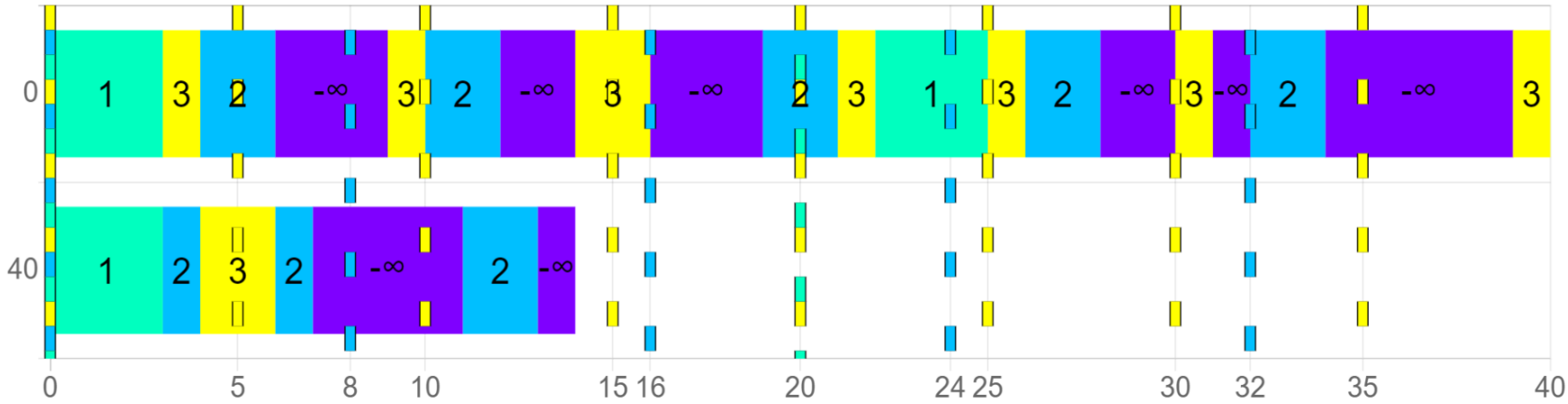


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	1	-Infinity	−
2	0	8	8	2	0	3	0	-Infinity	−
3	0	20	20	1	0	4	4	-Infinity	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	−

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

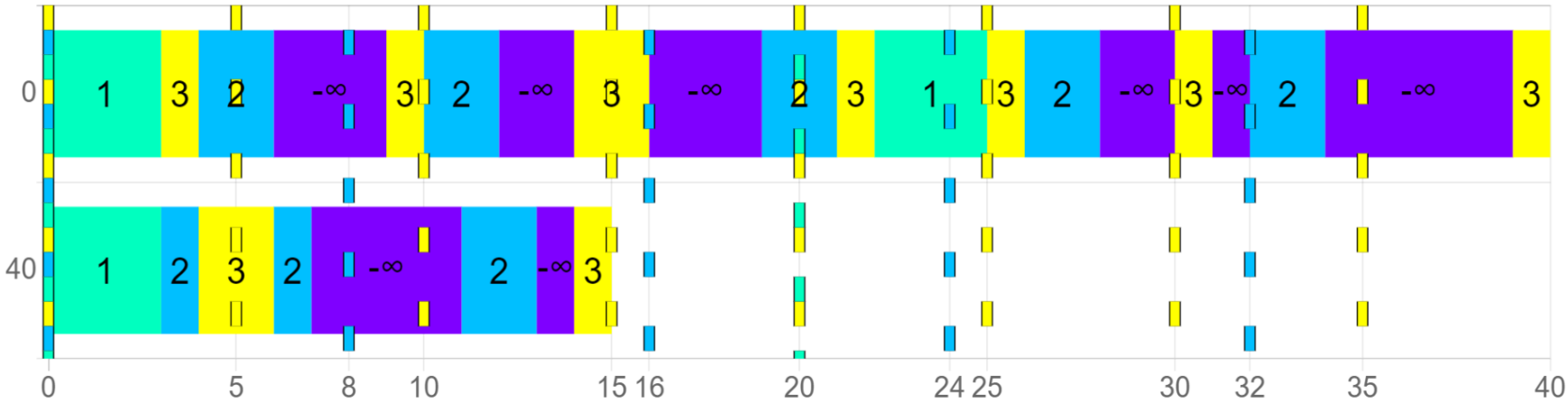


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	0	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

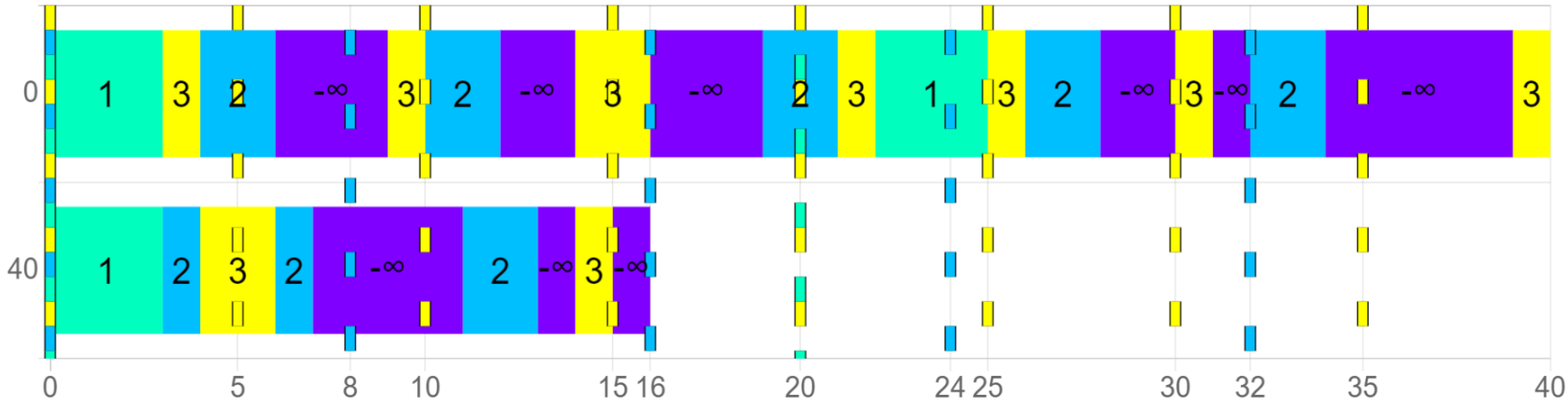


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	0	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

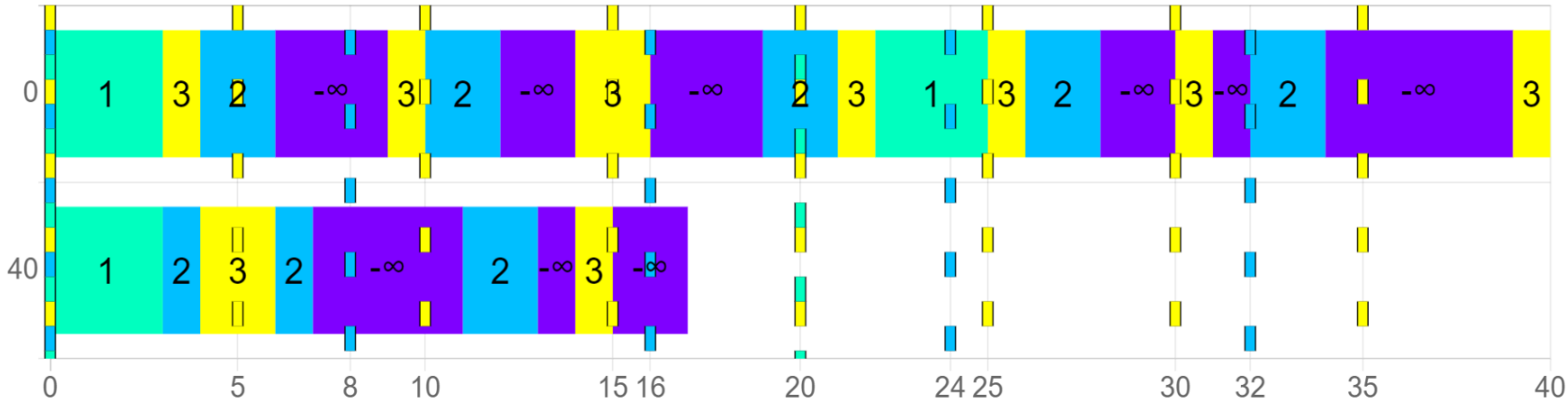


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	3	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

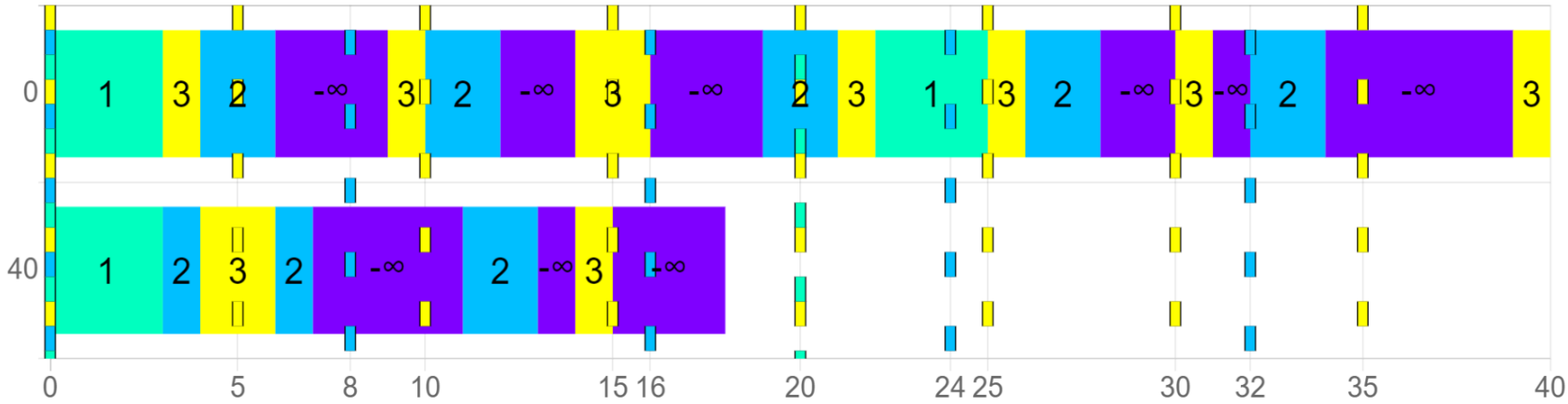


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	2	-Infinity	
2	0	8	8	2	2	3	2	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

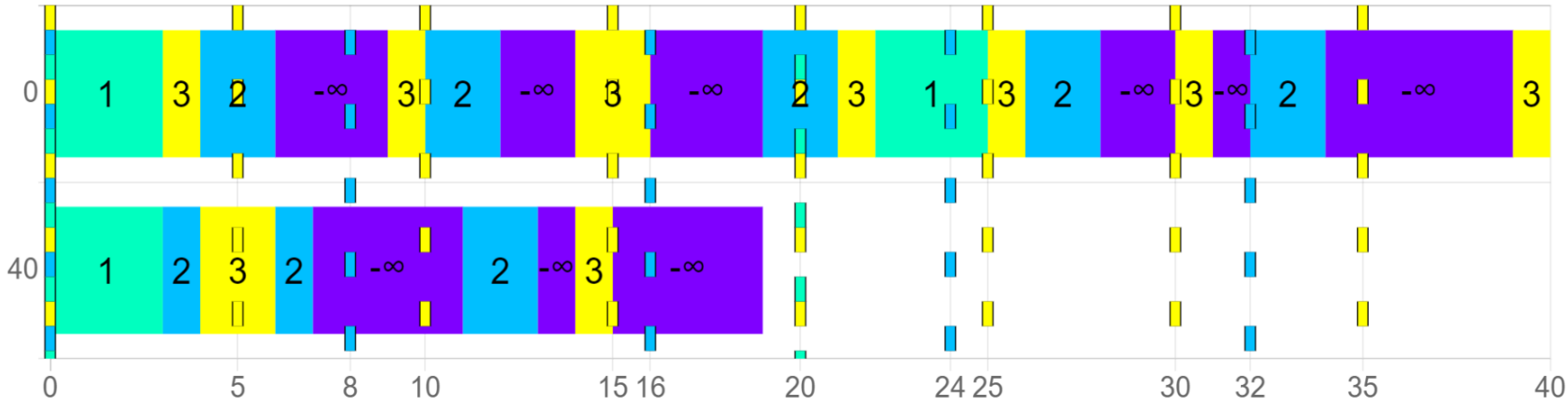


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	1	-Infinity	
2	0	8	8	2	2	3	1	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

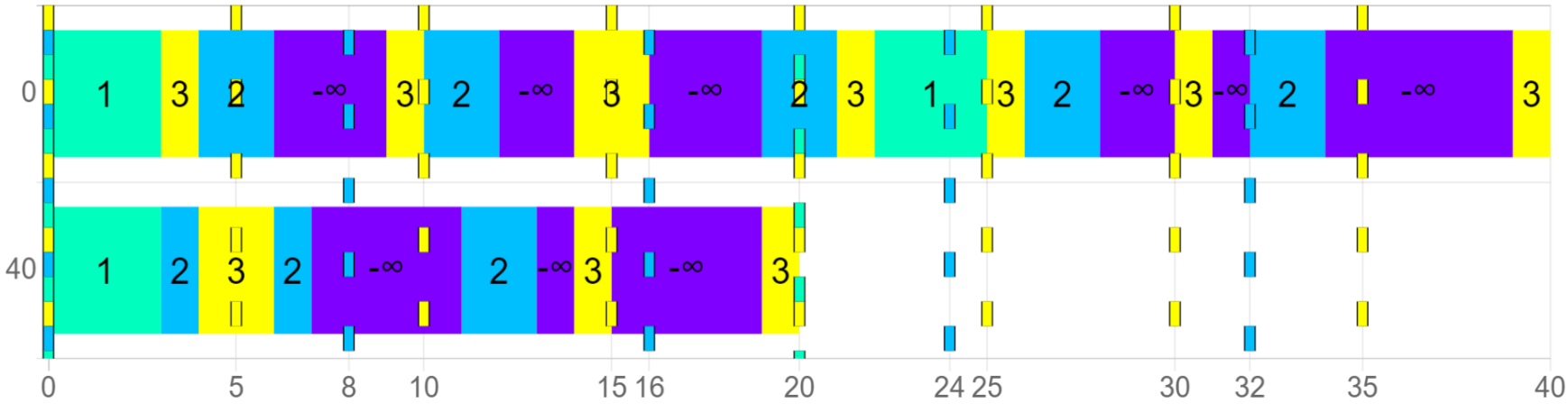


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	1	4	0	-Infinity	
2	0	8	8	2	2	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

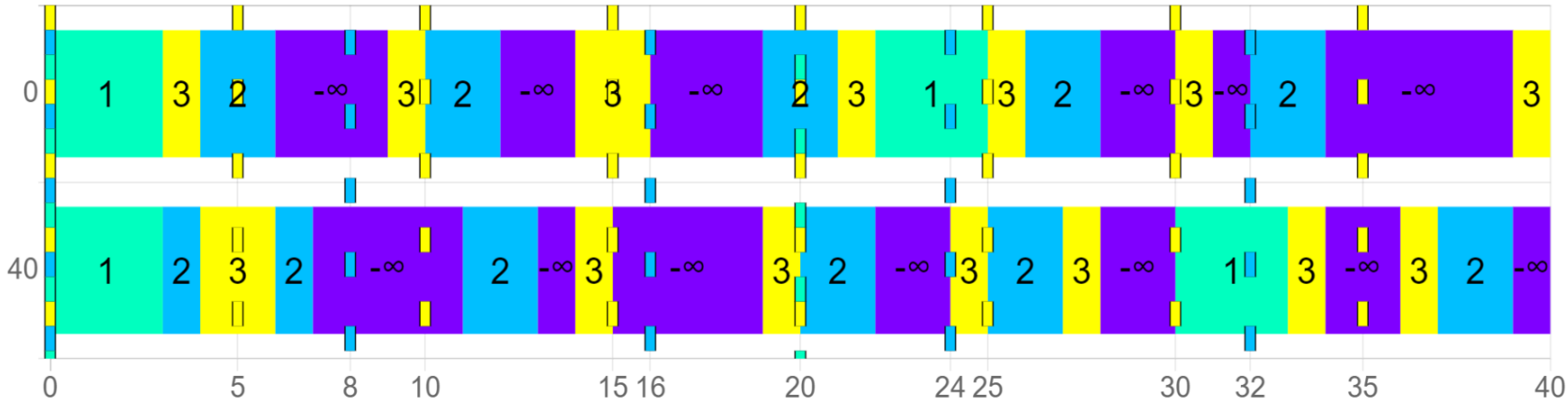


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	0	-Infinity	
2	0	8	8	2	2	3	0	-Infinity	
3	0	20	20	1	0	4	4	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

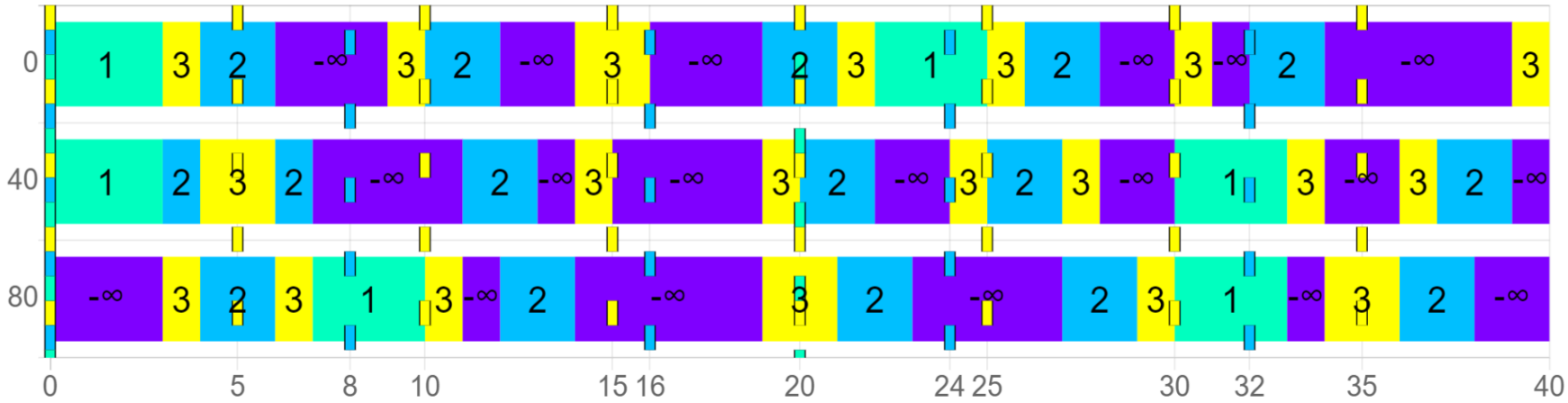


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	3	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	0	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

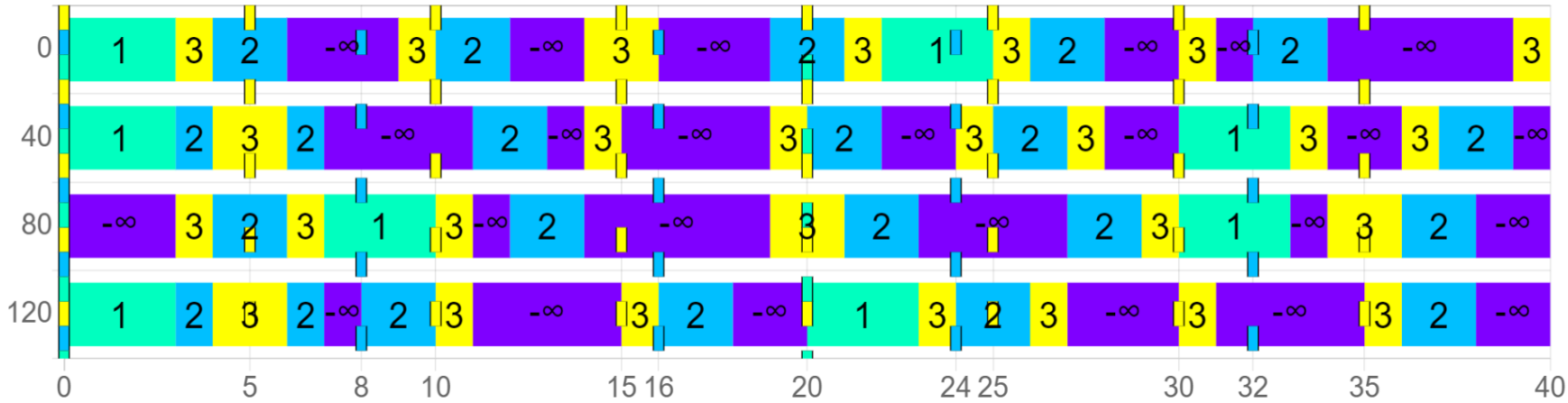


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	4	-Infinity	
2	0	8	8	2	0	3	1	-Infinity	
3	0	20	20	1	0	4	0	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

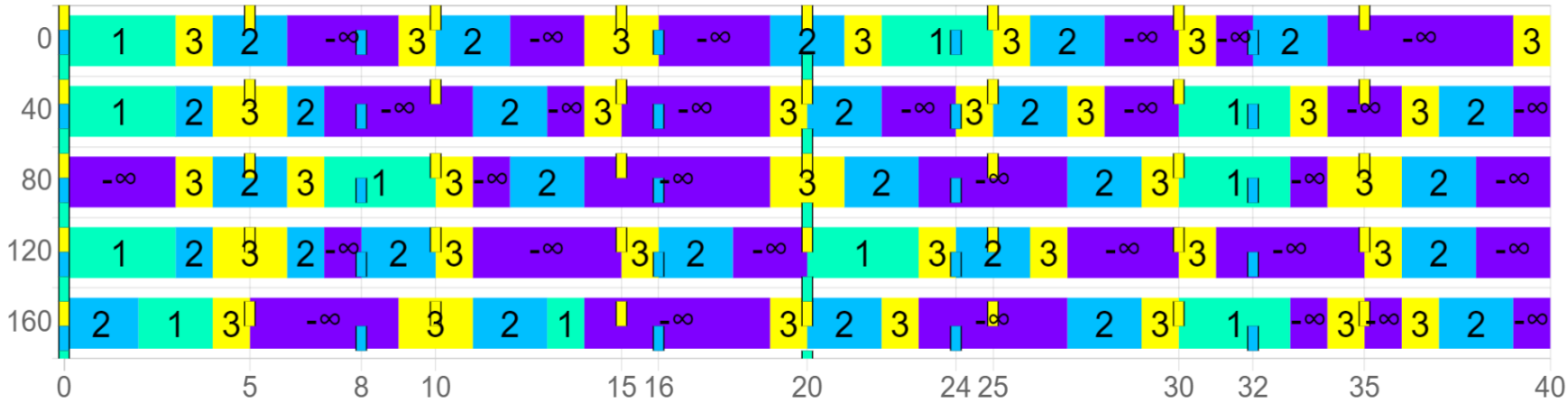


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	4	-Infinity	−
2	0	8	8	2	0	3	0	-Infinity	−
3	0	20	20	1	0	4	4	-Infinity	−
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	−

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*

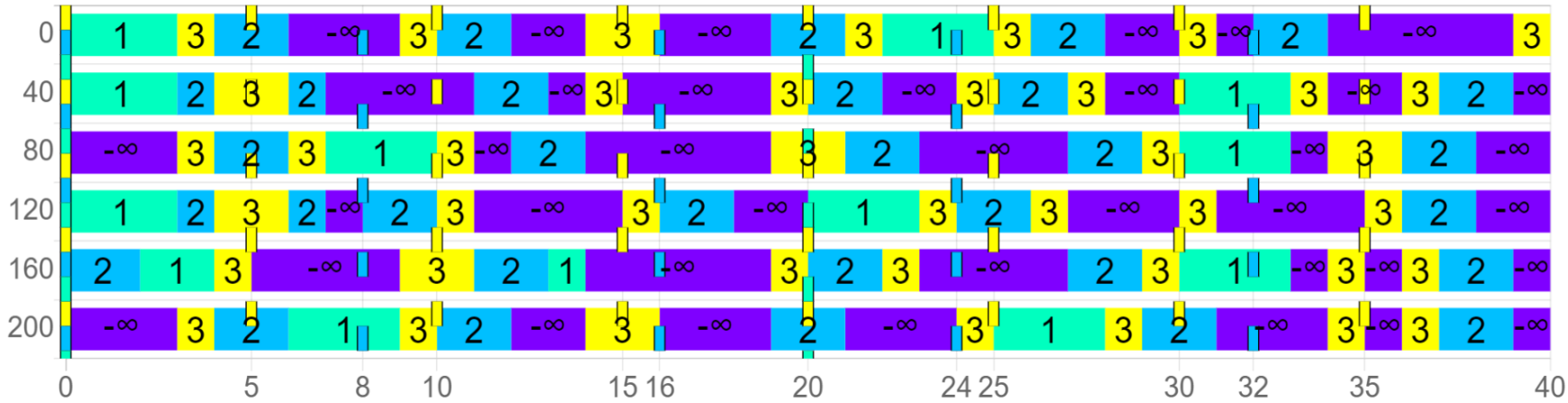


WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	3	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	0	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Scheduling Obfuscation: Verteidigung (?) durch “TaskShuffler”

- Zufälliges Umordnen der Tasks (**Schedule Randomization**)
  - ☞ *“Wie viel kann man die Ausführungsreihenfolge durcheinanderwürfeln, ohne dass Tasks ihre Deadlines verletzen?”*



WCET	arrivalTime	deadline	period	priority	remainingWork	ts invMax	ts invRem	ts minInvPrio	
1	0	5	5	3	0	4	3	-Infinity	
2	0	8	8	2	0	3	0	-Infinity	
3	0	20	20	1	0	4	1	-Infinity	
Infinity	0	Infinity	Infinity	-Infinity	Infinity	Infinity	Infinity	-Infinity	

https://www.simonlanger.com/uni/KvBK/

Restriktive Anforderungen für “Shufflen”

- (meist) **Einschränkung** auf Single-Core Architekturen und periodische Tasks
- **Overhead** durch häufigere Scheduleraufrufe und aufwendigere Schedulingentscheidungen
- **Verringerter Spielraum** bei unerwarteter (Not)Situation

Flexibilität ▼

Performance ▼

Kosten ▲

Restriktive Anforderungen für “Shufflen”

- (meist) **Einschränkung** auf Single-Core Architekturen und periodische Tasks
- **Overhead** durch häufigere Scheduleraufrufe und aufwendigere Schedulingentscheidungen
- **Verringerter Spielraum** bei unerwarteter (Not)Situation

Flexibilität ▼

Performance ▼

Kosten ▲

Restriktive Anforderungen für “Shufflen”

- (meist) **Einschränkung** auf Single-Core Architekturen und periodische Tasks
- **Overhead** durch häufigere Scheduleraufrufe und aufwendigere Schedulingentscheidungen
- **Verringerter Spielraum** bei unerwarteter (Not)Situation

Flexibilität ▼

Performance ▼

Kosten ▲

Restriktive Anforderungen für “Shufflen”

- (meist) **Einschränkung** auf Single-Core Architekturen und periodische Tasks
- **Overhead** durch häufigere Scheduleraufrufe und aufwendigere Schedulingentscheidungen
- **Verringerter Spielraum** bei unerwarteter (Not)Situation

Flexibilität ▼

Performance ▼

Kosten ▲

Restriktive Anforderungen für “Shufflen”

- (meist) **Einschränkung** auf Single-Core Architekturen und periodische Tasks
- **Overhead** durch häufigere Scheduleraufrufe und aufwendigere Schedulingentscheidungen
- **Verringerter Spielraum** bei unerwarteter (Not)Situation

Flexibilität ▼

Performance ▼

Kosten ▲

Stabilität für Sicherheit opfern?

Verschlimmerung der Konsequenzen bei Unterschätzung der Worst-Case Execution Time (WCET)

Angriffsfläche ▲

Provozieren eines **Deadline Miss** wird deutlich einfacher!

Inkompatibilität mit Features moderner Echtzeitbetriebssysteme (Real-Time Operating Systems)

Fehleranfälligkeit ▲

Deadlock-freie und Deadline-einhaltende Synchronisation wird enorm **erschwert**!

Stabilität für Sicherheit opfern?

Verschlimmerung der Konsequenzen bei Unterschätzung der Worst-Case Execution Time (WCET)

Angriffsfläche ▲

Provozieren eines **Deadline Miss** wird deutlich einfacher!

Inkompatibilität mit Features moderner Echtzeitbetriebssysteme (Real-Time Operating Systems)

Fehleranfälligkeit ▲

Deadlock-freie und Deadline-einhaltende Synchronisation wird enorm **erschwert**!

Stabilität für Sicherheit opfern?

Verschlimmerung der Konsequenzen bei Unterschätzung der Worst-Case Execution Time (WCET)

Angriffsfläche ▲

Provozieren eines **Deadline Miss** wird deutlich einfacher!

Inkompatibilität mit Features moderner Echtzeitbetriebssysteme (Real-Time Operating Systems)

Fehleranfälligkeit ▲

Deadlock-freie und Deadline-einhaltende Synchronisation wird enorm **erschwert**!

Abschreckung bedeutet nicht Schutz!

Fehlen einer (universell) anerkannten Metrik

Überbewertung “Shuffle”-Schutzwirkung

Gefahr: Verzicht auf effektive Schutzmaßnahmen!

Perfektes zufälliges Umordnen (mehr unterschiedliche Situationen) gar nicht wünschenswert?!

Schaffung neuer Angriffsflächen

Gefahr: Vom Angreifer gewünschte Situation tritt (ziemlich sicher) irgendwann ein!

Abschreckung bedeutet nicht Schutz!

Fehlen einer (universell) anerkannten Metrik

Überbewertung “Shuffle”-Schutzwirkung

Gefahr: Verzicht auf effektive Schutzmaßnahmen!

Perfektes zufälliges Umordnen (mehr unterschiedliche Situationen) gar nicht wünschenswert?!

Schaffung neuer Angriffsflächen

Gefahr: Vom Angreifer gewünschte Situation tritt (ziemlich sicher) irgendwann ein!

Abschreckung bedeutet nicht Schutz!

Fehlen einer (universell) anerkannten Metrik

Überbewertung “Shuffle”-Schutzwirkung

Gefahr: Verzicht auf effektive Schutzmaßnahmen!

Perfektes zufälliges Umordnen (mehr unterschiedliche Situationen) gar nicht wünschenswert?!

Schaffung neuer Angriffsflächen

Gefahr: Vom Angreifer gewünschte Situation tritt (ziemlich sicher) irgendwann ein!

Isolation schlägt Obfuscation?!

Komponente Zeit zentral für Echt**zeit**systeme: (vermeidbares) “Spiel mit dem Feuer”!

Flexibilität, Performance ▼

Kosten, Angriffsfläche, Fehleranfälligkeit ▲

Überbewertung “Shuffle”-Schutzwirkung

Schaffung neuer Angriffsflächen

Alternativen

- ☞ Temporale (zeitliche) und lokale (physikalische) **Kapselung**
- ☞ **Speicherschutz** (vgl. SP1, SP2)
- ☞ Einführung eines Berechtigungssystems (**Permission System**)

Isolation schlägt Obfuscation?!

Komponente Zeit zentral für Echt**zeit**systeme:
(vermeidbares)
“Spiel mit dem Feuer”!

Flexibilität, Performance ▼

Kosten, Angriffsfläche, Fehleranfälligkeit ▲

Überbewertung “Shuffle”-Schutzwirkung

Schaffung neuer Angriffsflächen

Alternativen

- ☞ Temporale (zeitliche) und lokale (physikalische) **Kapselung**
- ☞ **Speicherschutz** (vgl. SP1, SP2)
- ☞ Einführung eines Berechtigungssystems (**Permission System**)

Isolation schlägt Obfuscation?!

Komponente Zeit zentral für Echt**zeit**systeme: (vermeidbares) “Spiel mit dem Feuer”!

Flexibilität, Performance ▼

Kosten, Angriffsfläche, Fehleranfälligkeit ▲

Überbewertung “Shuffle”-Schutzwirkung

Schaffung neuer Angriffsflächen

Alternativen

- ☞ Temporale (zeitliche) und lokale (physikalische) **Kapselung**
- ☞ **Speicherschutz** (vgl. SP1, SP2)
- ☞ Einführung eines Berechtigungssystems (**Permission System**)

Isolation schlägt Obfuscation?!

Komponente Zeit zentral für Echt**zeit**systeme: (vermeidbares) “Spiel mit dem Feuer”!

Flexibilität, Performance ▼

Kosten, Angriffsfläche, Fehleranfälligkeit ▲

Überbewertung “Shuffle”-Schutzwirkung

Schaffung neuer Angriffsflächen

Alternativen

- ☞ Temporale (zeitliche) und lokale (physikalische) **Kapselung**
- ☞ **Speicherschutz** (vgl. SP1, SP2)
- ☞ Einführung eines Berechtigungssystems (**Permission System**)


Chang Liu and James W. Layland. “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”. In: *J. ACM* 20 (1973), pp. 46–61.


Chien-Ying Chen et al. “A Novel Side-Channel in Real-Time Schedulers”. In: *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2019), pp. 90–102.


Chien-Ying Chen et al. “On Scheduler Side-Channels in Dynamic-Priority Real-Time Systems”. In: *ArXiv* abs/2001.06519 (2020).


Man-Ki Yoon et al. “TaskShuffler: A Schedule Randomization Protocol for Obfuscation against Timing Inference Attacks in Real-Time Systems”. In: *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2016), pp. 1–12.


Man-Ki Yoon et al. “TaskShuffler++: Real-Time Schedule Randomization for Reducing Worst-Case Vulnerability to Timing Inference Attacks”. In: *ArXiv* abs/1911.07726 (2019).


Mitra Nasri et al. “On the Pitfalls and Vulnerabilities of Schedule Randomization Against Schedule-Based Attacks”. In: *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2019), pp. 103–116.


Bryan C. Ward et al. “Security Considerations for Next-Generation Operating Systems for Cyber-Physical Systems”. In: MIT Lincoln Laboratory. 2019.