Breaking and Securing Memory Isolation in Texas Instruments Microcontrollers

Marton Bognar, Jo Van Bulck

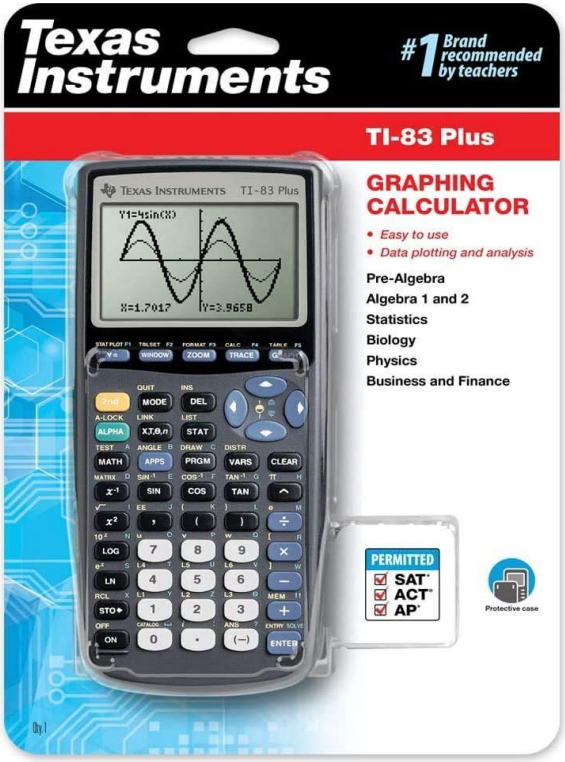
(based on work with Cas Magnus and Frank Piessens)

DistriNet, KU Leuven, Belgium

DistriNet

KU LEUVEN

Texas Instruments



Texas Instruments

Texas Instruments

Wikipedia
The Free Encyclopedia

Search Wikipedia | Search

Texas Instruments

51 languages

Article | Talk | Read | Edit | View history | Tools

From Wikipedia, the free encyclopedia

Coordinates: 32.9110°N 96.7523°W

Contents hide

Texas Instruments Incorporated (**TI**) is an American multinational semiconductor company headquartered in Dallas, Texas.[5] It is one of the top 10 semiconductor companies worldwide based on sales volume.[6] The company's focus is on developing analog chips and embedded processors, which account for more than 80% of its revenue.[7] TI also produces digital light processing (DLP) technology and education technology[7] products including calculators, microcontrollers, and multi-core processors.[8]

Texas Instruments emerged in 1951 after a reorganization of Geophysical Service Incorporated, a company founded in 1930 that manufactured equipment for use in the seismic industry, as well as defense electronics.[9] TI produced the world's first commercial silicon transistor in 1954,[10] and the same year designed and manufactured the first transistor radio. Jack Kilby invented the integrated circuit in 1958 while working at TI's Central Research Labs. TI also invented the hand-held calculator in 1967, and introduced the first single-chip microcontroller in 1970, which combined all the elements of computing onto one piece of silicon.[11]

In 1987, TI invented the digital light processing device (also known as the DLP chip), which serves as the foundation for the company's DLP technology and DLP Cinema.[11] TI released the popular TI-81 calculator in 1990, which made it a leader in the graphing calculator industry. Its defense business was sold to Raytheon Company in 1997; this allowed TI to strengthen its focus on digital solutions.[12] After the acquisition of National Semiconductor in 2011, the company had a combined portfolio of 45,000 analog products and customer design tools.[13] In the stock market, Texas Instruments is often regarded as an indicator for the semiconductor and electronics industry as a whole, since the company sells to more than 100,000 customers.[14][15][16]

Texas Instruments Incorporated

Sign at TI's Dallas headquarters

Company type	Public
Traded as	Nasdaq: TXN Nasdaq-100 component S&P 100 component S&P 500 component
Industry	Semiconductors
Predecessor	Geophysical Service
Founded	1930; 95 years ago (as

Texas Instruments

of 1954, at the IRE off-the-record conference on solid-state devices, and was later published in the *Journal of Applied Physics*. Working independently in April 1954, Gordon Teal at TI created the first commercial silicon transistor and tested it on April 14, 1954. On May 10, 1954, at the Institute of Radio Engineers National Conference on Airborne Electronics in Dayton, Ohio, Teal presented a paper: "Some Recent Developments in Silicon and Germanium Materials and Devices".[25]

Texas Instruments

of 1954, at the IRE off-the-record conference on solid-state devices, and was later published in the *Journal of Applied Physics*. Working independently in April 1954, Gordon Teal at TI created the first commercial silicon transistor and tested it on April 14, 1954. On May 10, 1954, at the Institute of Radio Engineers National Conference on Airborne Electronics in Dayton, Ohio, Teal presented a paper: "Some Recent Developments in Silicon and Germanium Materials and Devices".[25]

Jack Kilby, an employee at TI, invented the integrated circuit in 1958.[26] Kilby recorded his initial ideas concerning the integrated circuit in July 1958, and successfully demonstrated the world's first working integrated circuit on September 12, 1958.[27] Six months later, Robert Noyce of Fairchild Semiconductor (who went on to co-found Intel) independently developed

Texas Instruments

Texas Instruments

In 1964, TI began development of the first laser guidance system for precision-guided munitions, leading to the Paveway series of laser-guided bombs (LGBs). The first LGB was the BOLT-117.[54]

About us

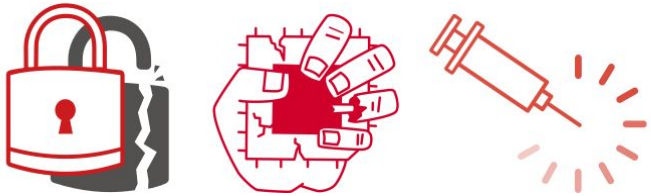
- **Marton Bognar:** PhD candidate @ KU Leuven, Belgium
  - PhD thesis: “Security Arms Race at the Hardware-Software Boundary”
  - https://mici.hu/

- **Jo Van Bulck:** Professor @ KU Leuven, Belgium
  - Microarchitectural Side-Channel Attacks and Defenses
  - Trusted Execution Environments
  - https://vanbulck.net/

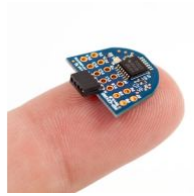
About DistriNet systems security research

- Trust across the **system stack:** *App > compiler > OS > CPU > μ-arch*
- Integrated **attack-defense** perspective and **open-source** prototypes

Side-channel analysis
(SGX-Step, AEX-Notify)



Transient-execution attacks
(Intel x86 SGX)

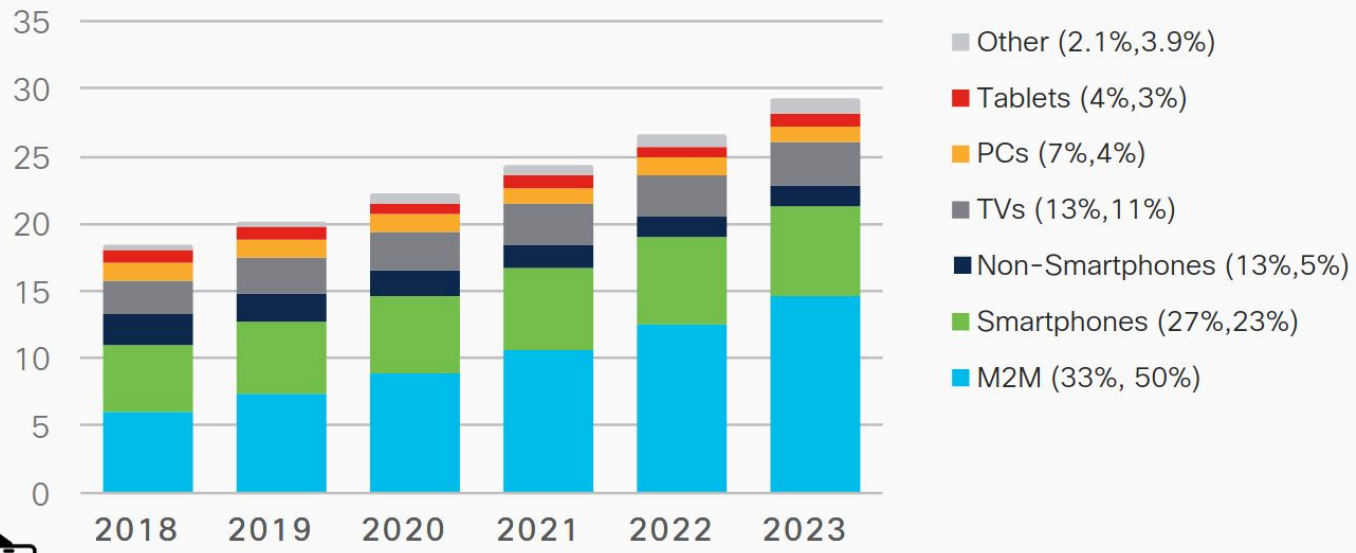


Embedded trust
(MSP430, Sancus)

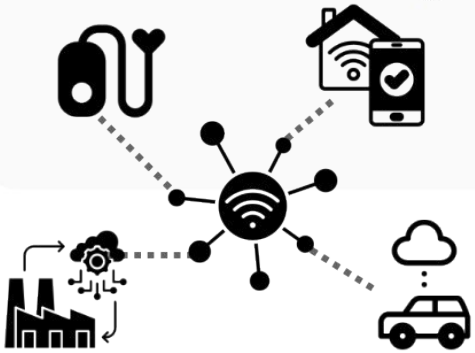
Part I: Why should we care?

Context: Growth of the Internet of Things (IoT)

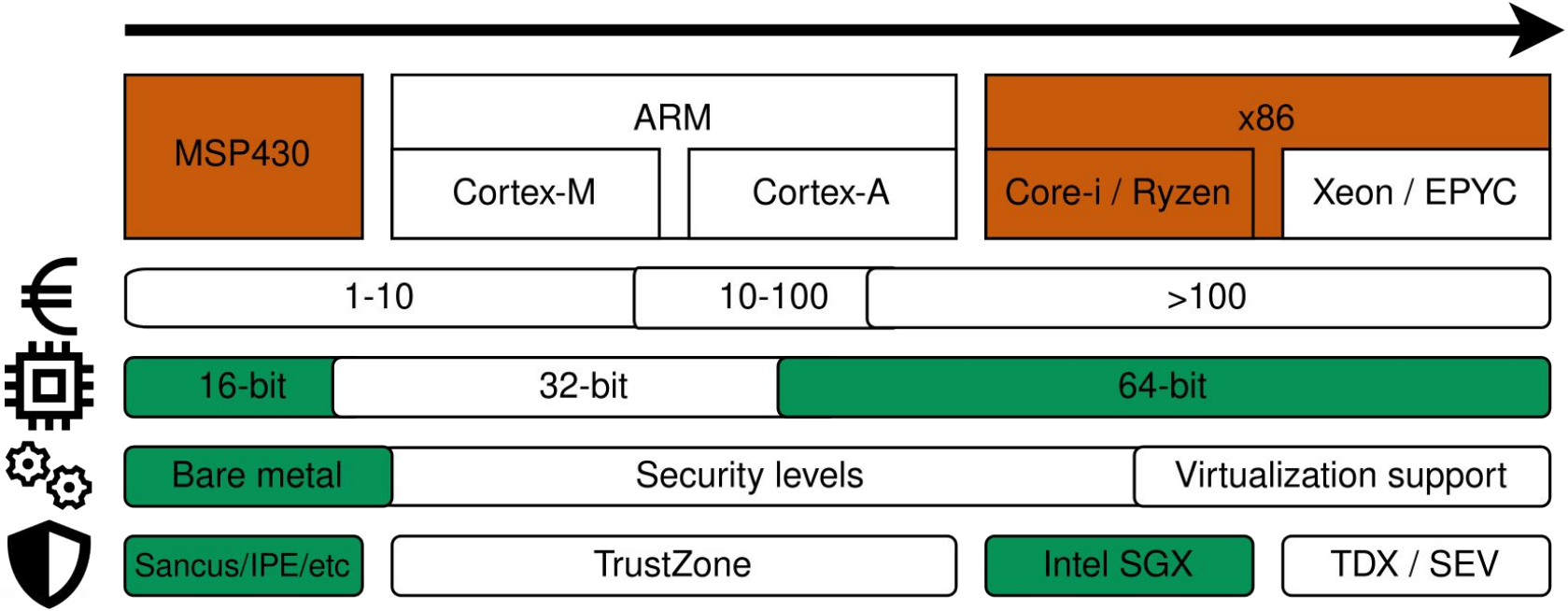
10% CAGR
2018–2023

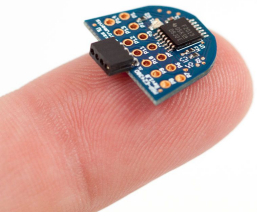


* Figures (n) refer to 2018, 2023 device share

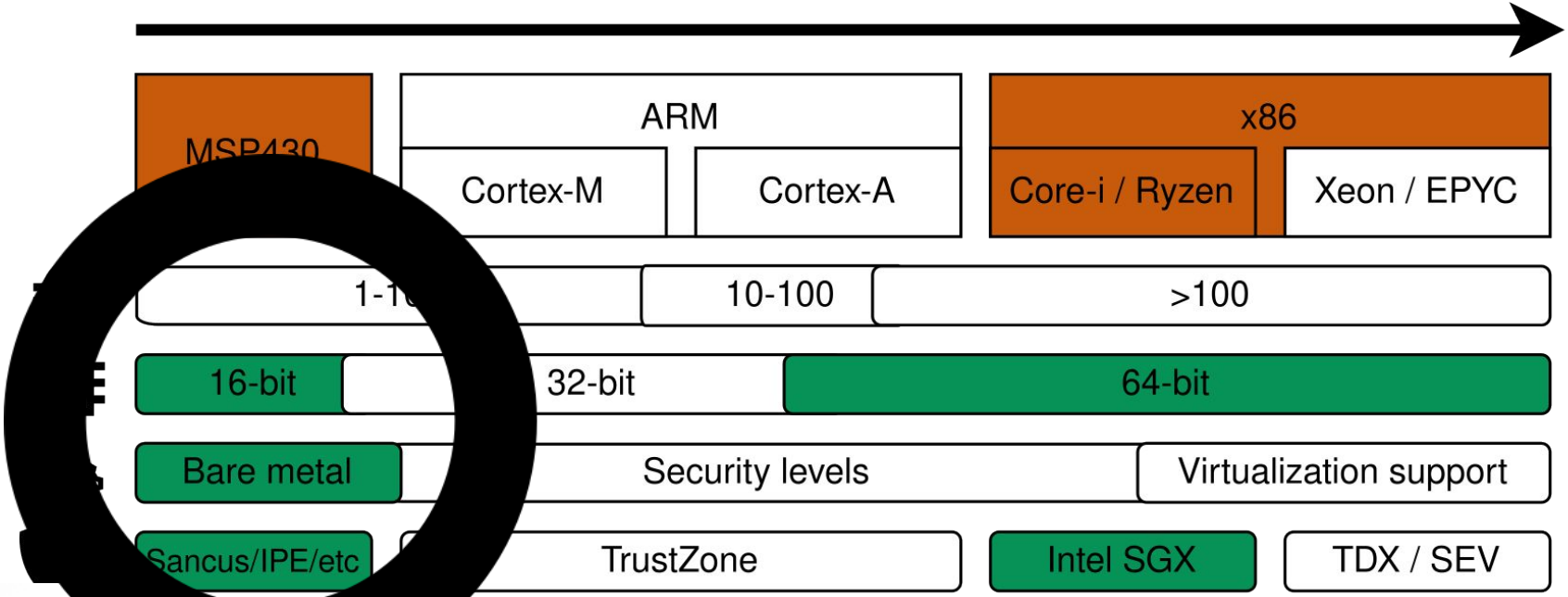


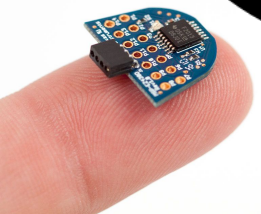
Computing spectrum: “Low-end” vs. “high-end”





Computing spectrum: “Low-end” vs. “high-end”



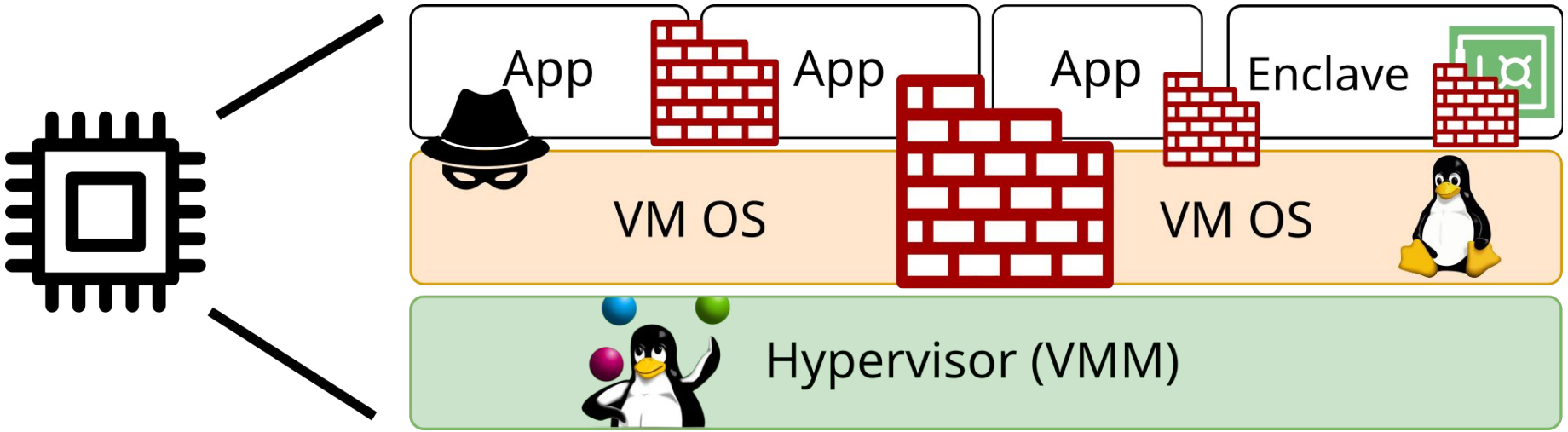


Heterogenous & underexplored!

"Embedded-systems security is, for lack of a better word, a mess."

– John Viega & Hugh Thompson (S&P'12)

Memory isolation: Conventional “high-end” systems

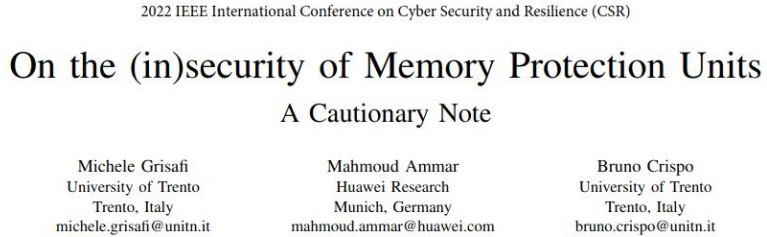


- Software **protection domains:** Processes, VMs, enclaves
- CPU support **memory isolation:** Virtual memory + privilege rings

Memory isolation: “Low-end” microcontrollers

- **Often: None** 🤦
  - 8/16-bit *single address space*
  - No virtual memory
  - No operating system

- **Memory protection unit (MPU):**
  - Goal: *bug detection ≠ security*
  - Misconfiguration
  - Attacker can often reconfigure it



2022 IEEE International Conference on Cyber Security and Resilience (CSR)

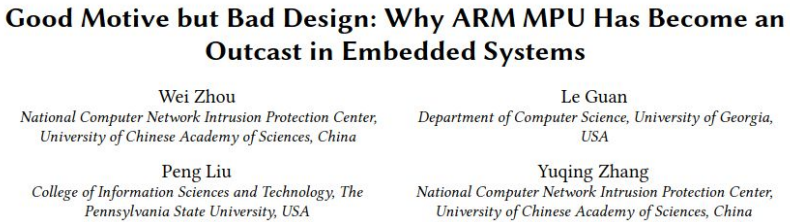
On the (in)security of Memory Protection Units

A Cautionary Note

Michele Grisafi
University of Trento
Trento, Italy
michele.grisafi@unitn.it

Mahmoud Ammar
Huawei Research
Munich, Germany
mahmoud.ammar@huawei.com

Bruno Crispo
University of Trento
Trento, Italy
bruno.crispo@unitn.it



Good Motive but Bad Design: Why ARM MPU Has Become an Outcast in Embedded Systems

Wei Zhou
National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, China

Le Guan
Department of Computer Science, University of Georgia, USA

Peng Liu
College of Information Sciences and Technology, The Pennsylvania State University, USA

Yuqing Zhang
National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, China

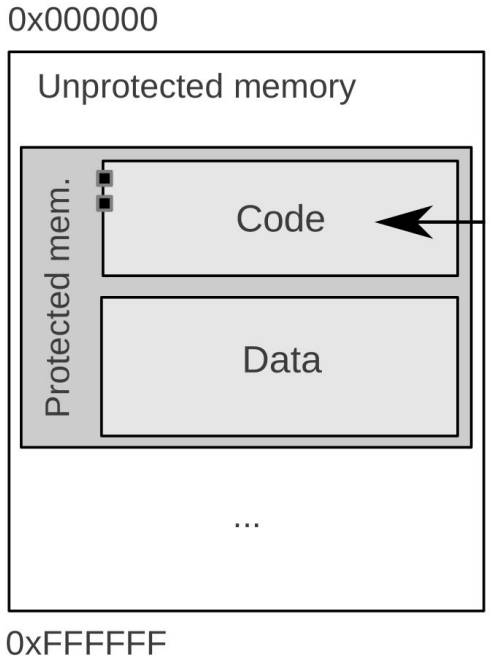
Emerging solutions: Embedded trusted computing

Embedded enclaved execution:

- Isolation & attestation
- Save + clear CPU state on interrupt

Small CPU (**openMSP430**):

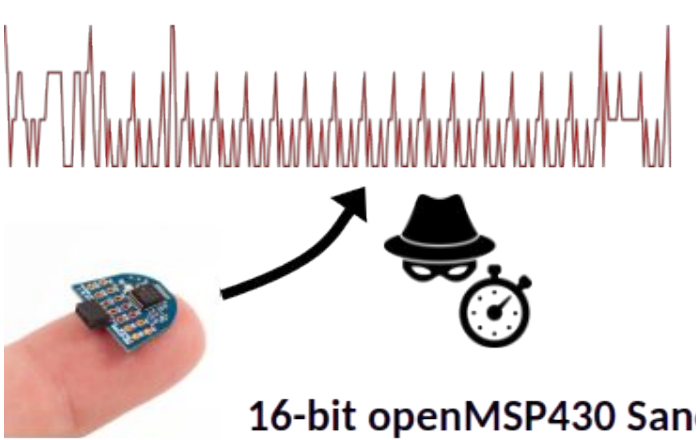
- Area: ≤ 2 kLUTs
- Deterministic execution: no pipeline/cache/MMU/…
- Research vehicle for rapid prototyping of attacks & mitigations



https://github.com/sancus-tee
https://downloads.distrinet-research.be/software/sancus

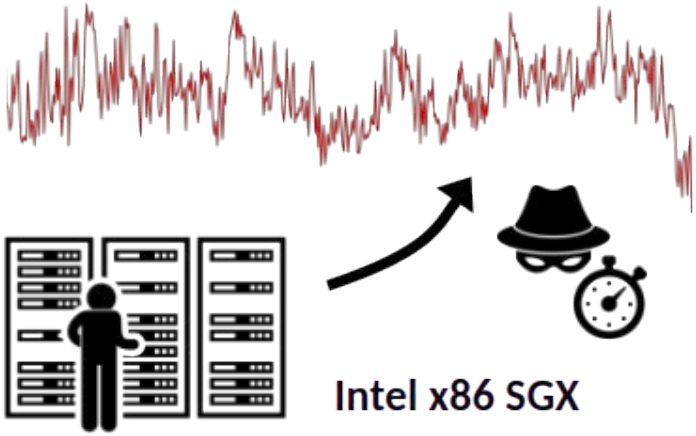
Noorman et al. Sancus 2.0: A Low-Cost Security Architecture for IoT devices. TOPS, 2017.

Synergy: Attacks on low-end and high-end TEEs



16-bit openMSP430 Sancus

- Small CPU, open source



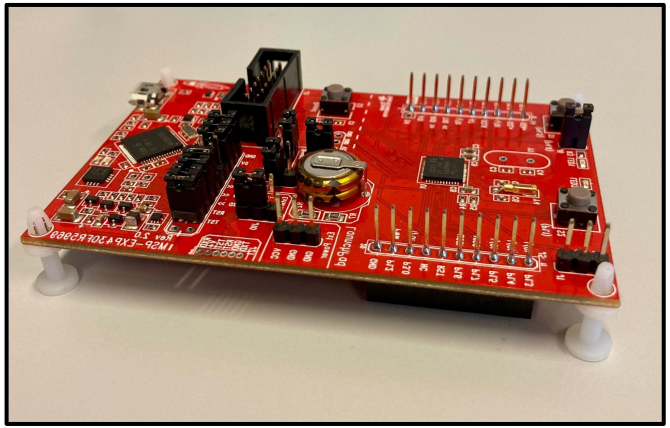
Intel x86 SGX

- Large CPU, proprietary

Van Bulck et al. “Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic”, CCS 2018.

Part II: Security analysis

Texas Instruments MSP430 microcontroller

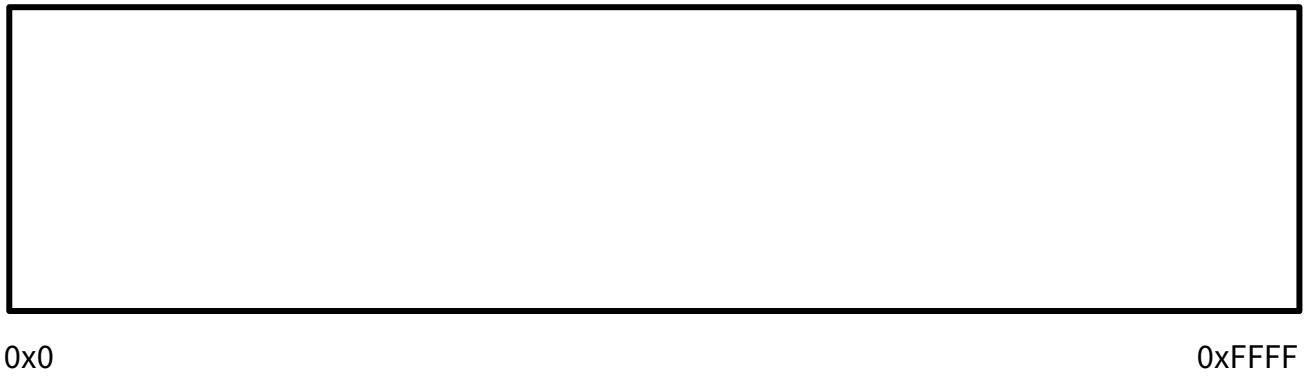


- Low-power microcontrollers
- FRAM edition (2014) with <u>security features:</u>
  - Physical tamper protection
  - Hardware AES cryptographic unit
  - Memory protection unit (MPU)
  - **Intellectual Property Encapsulation (IPE)**

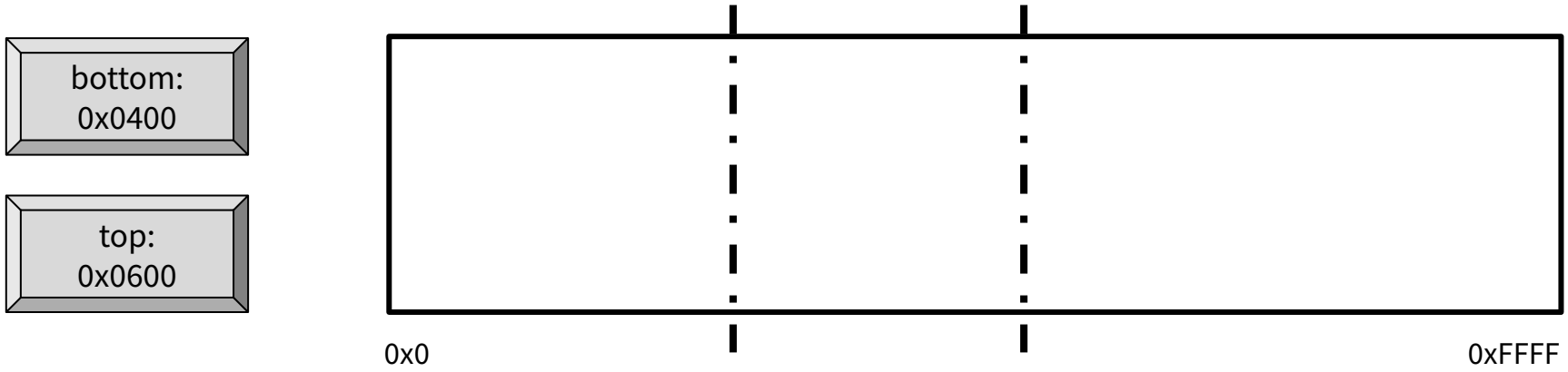
TI Embedded Security Portfolio –
Security is hard, TI makes it easier

"The IPE module protects a programmed portion of memory from read or write access from anywhere outside of the IP Encapsulated area, even by JTAG. This IPE module minimizes risk of exposure of critical or proprietary software from the rest of the application […]"

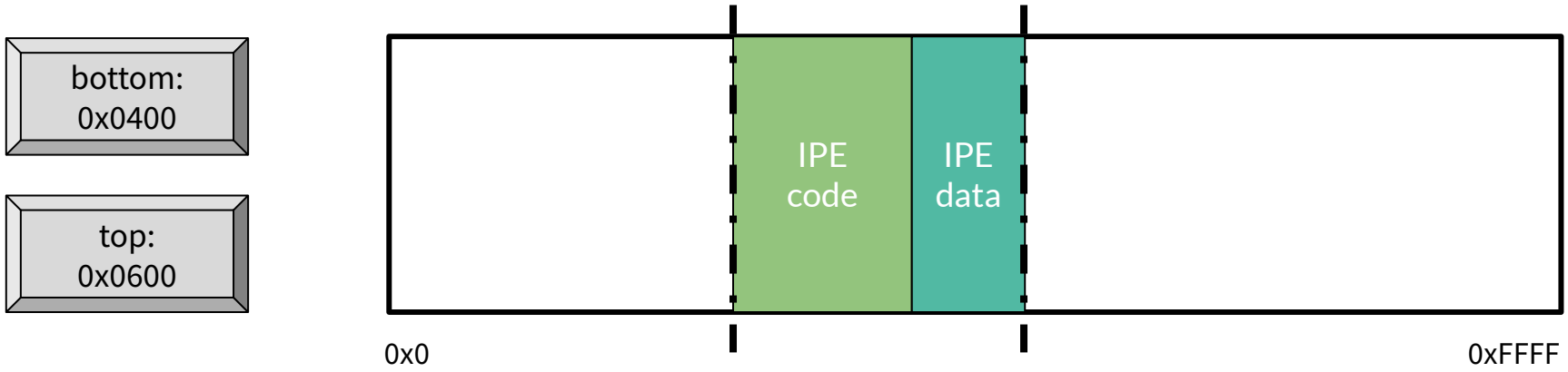
Intellectual Property Encapsulation



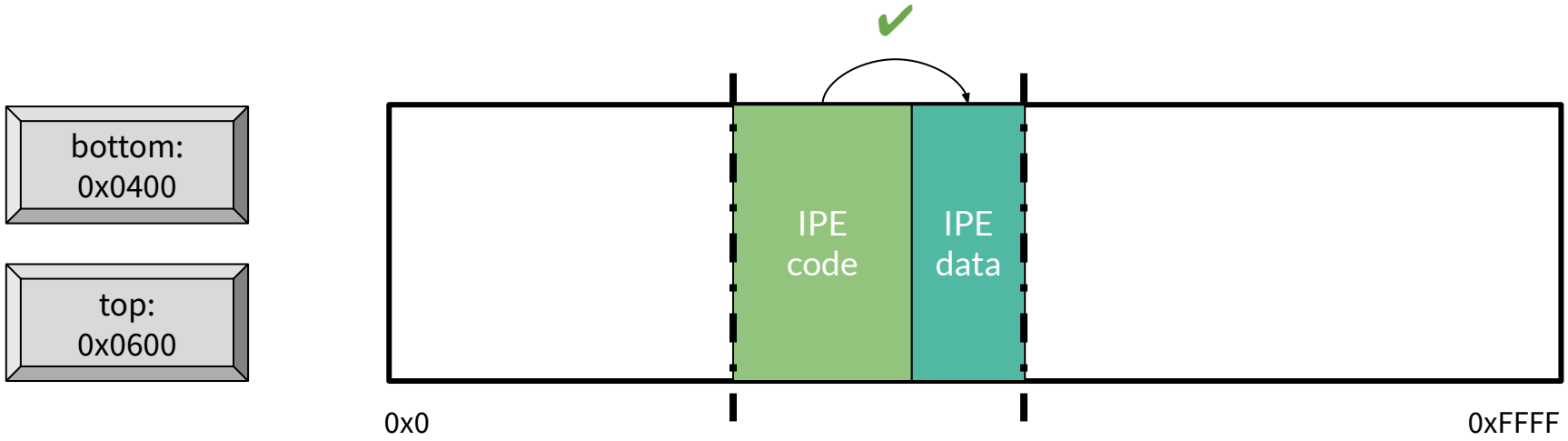
Intellectual Property Encapsulation



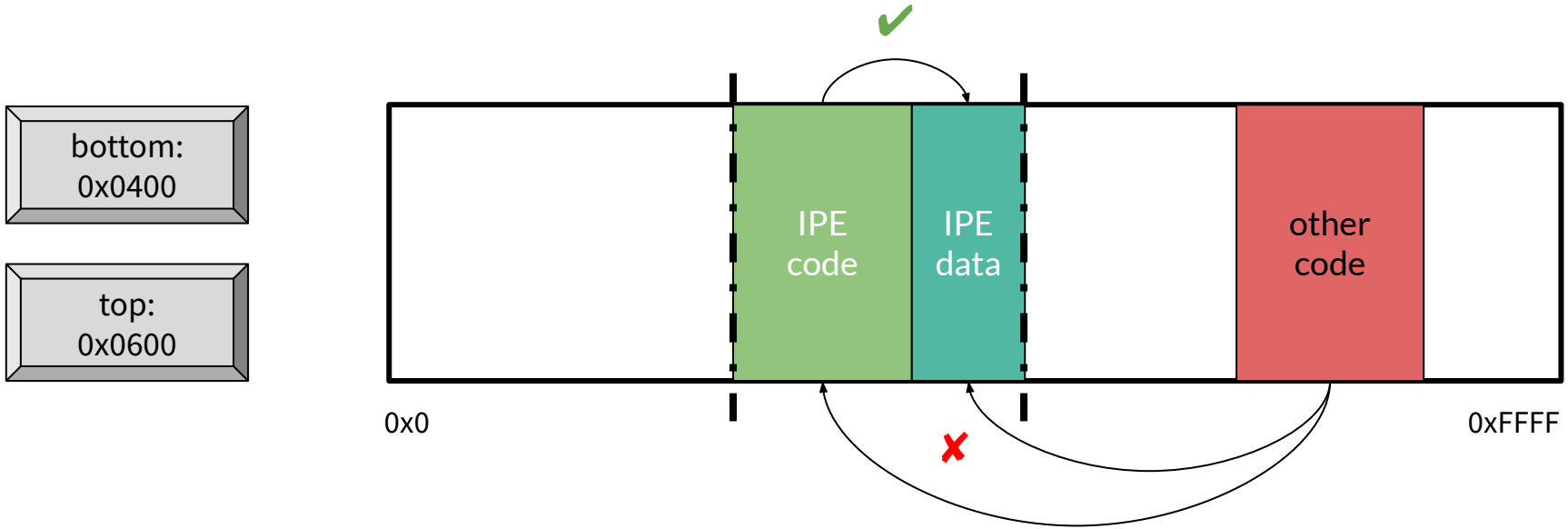
Intellectual Property Encapsulation



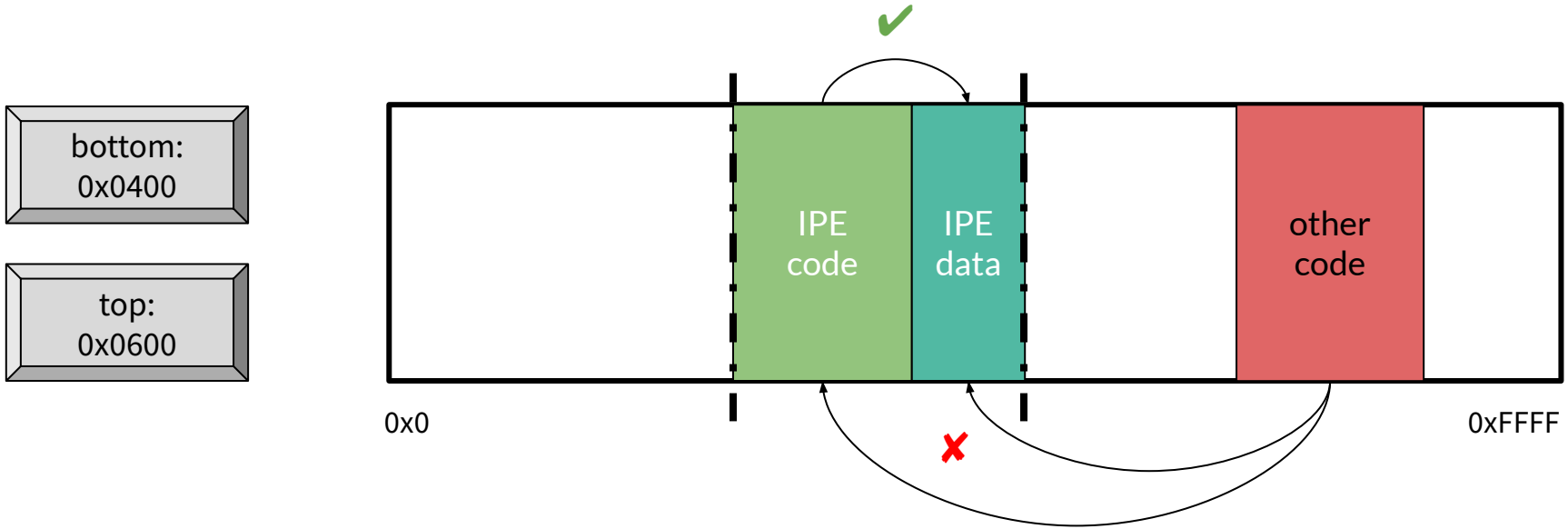
Intellectual Property Encapsulation



Intellectual Property Encapsulation

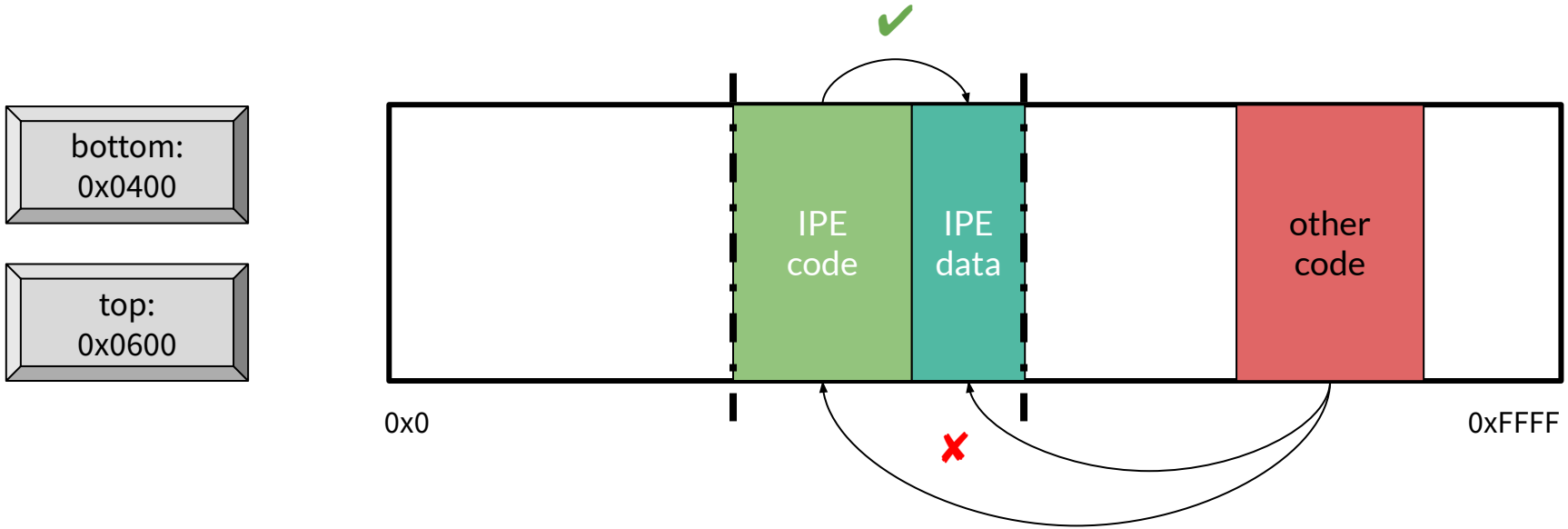


Intellectual Property Encapsulation



+ protection from JTAG debug port, direct memory access (DMA)

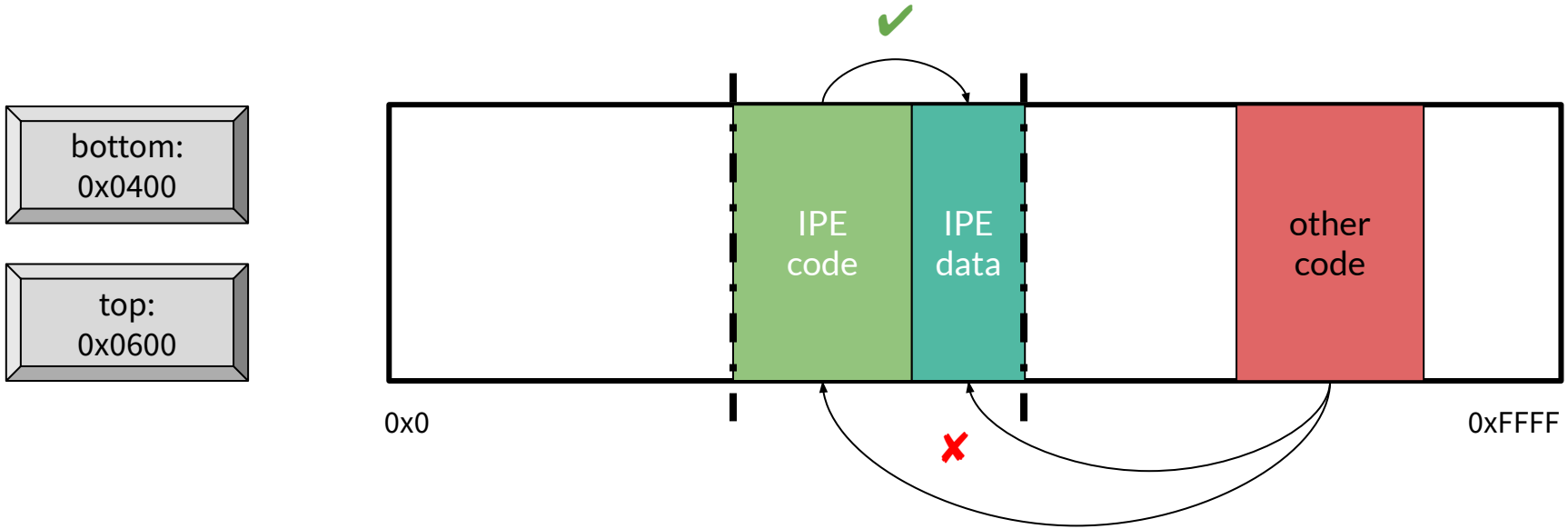
Intellectual Property Encapsulation



+ protection from JTAG debug port, direct memory access (DMA)

→ Program-counter-based access control

Intellectual Property Encapsulation



+ protection from JTAG debug port, direct memory access (DMA)

→ Program-counter-based access control

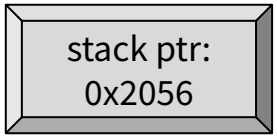
→ Looks like a **trusted execution environment (TEE)!**

Controlled *call* corruption

```
int factorial(int n) {
  int sub = n - 1;
  return (n * factorial(sub));
}

int main() {
  int result = factorial(5);
  result += 4;
}
```

Controlled *call* corruption

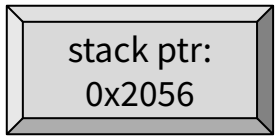



```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
stack:
2054: 0
2056: 0
2058: 0xBEEF
```

```
factorial:
6080: mov r12, r13
6082: sub #1, r12
6084: ...
```

Controlled *call* corruption

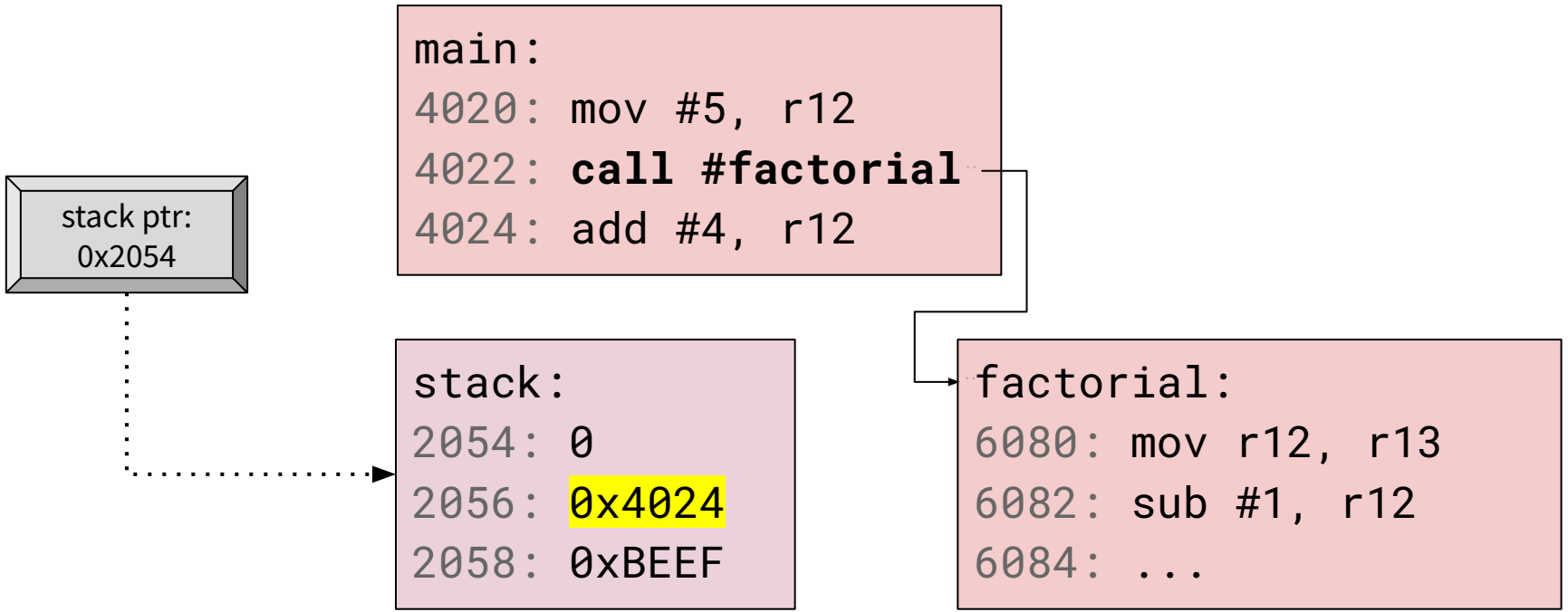



```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

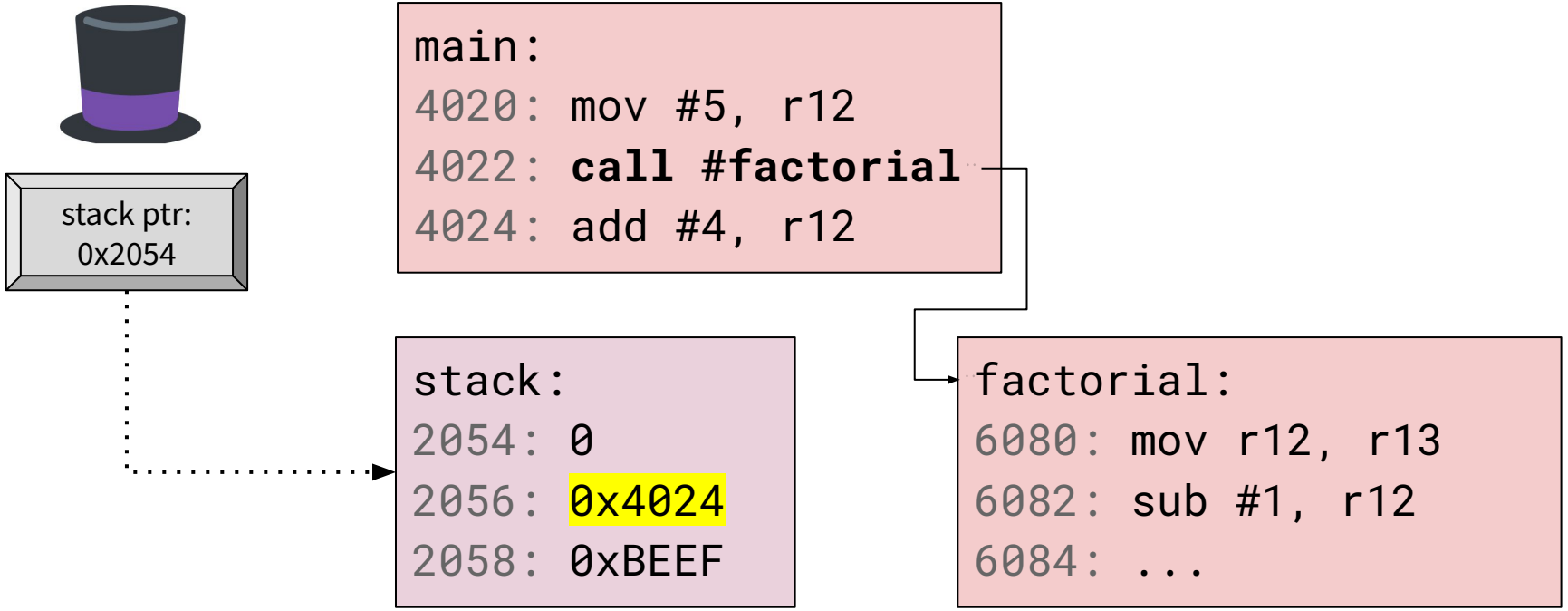
```
stack:
2054: 0
2056: 0
2058: 0xBEEF
```

```
factorial:
6080: mov r12, r13
6082: sub #1, r12
6084: ...
```

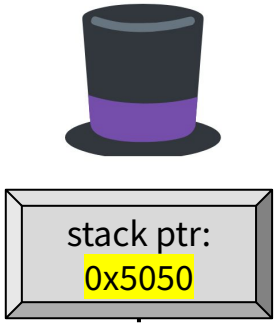
Controlled *call* corruption



Controlled *call* corruption



Controlled *call* corruption




```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: AES_KEY
5052: 0
5054: 0x42
```

```
factorial:
6080: mov r12, r13
6082: sub #1, r12
6084: ...
```

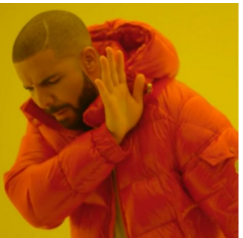
Controlled *call* corruption

stack ptr:
0x5050

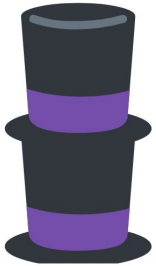
```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: AES_KEY
5052: 0
5054: 0x42
```

```
factorial:
6080: mov r12, r13
6082: sub #1, r12
6084: ...
```



Controlled *call* corruption

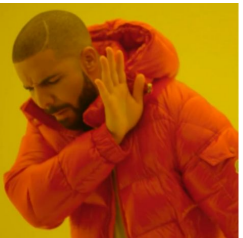


stack ptr:
0x5050

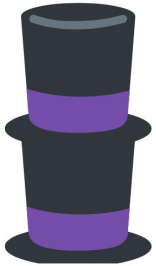
```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: AES_KEY
5052: 0
5054: 0x42
```

```
factorial:
6080: mov r12, r13
6082: sub #1, r12
6084: ...
```



Controlled *call* corruption



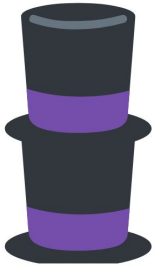
stack ptr:
0x5050

```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: 0
5052: 0
5054: 0x42
```

```
factorial_ipe:
5000: mov r12, r13
5002: sub #1, r12
5004: ...
```

Controlled *call* corruption

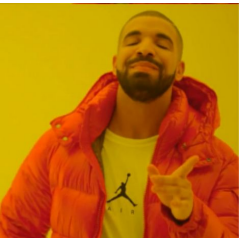


stack ptr:
0x504E

```
main:
4020: mov #5, r12
4022: call #factorial
4024: add #4, r12
```

```
ipe_data:
5050: 0x4024
5052: 0
5054: 0x42
```

```
factorial_ipe:
5000: mov r12, r13
5002: sub #1, r12
5004: ...
```



Controlled *call* corruption

Corrupt code in IPE to crash the application

Controlled *call* corruption

Corrupt code in IPE to crash the application

Overwrite secret data with known values

Controlled *call* corruption

Corrupt code in IPE to crash the application

Overwrite secret data with known values

Insert a universal read gadget

Controlled *call* corruption

Table 3: A subset of instruction encodings on MSP430, falling in the RAM (*italic*) or FRAM ranges (cf. Table 2).

Encoding	Inst	Encoding	Inst
0xFxxx	and	0x8xxx	sub
0xExxx	xor	0x7xxx	subc
0xDxxx	bis	0x6xxx	addc
0xCxxx	bic	0x5xxx	add
0xBxxx	bit	0x4xxx	mov
0xAxxx	dadd	*0x3xxx*	*jmp, jl, jge, jn*
0x9xxx	cmp	*0x2xxx*	*jc, jnc, jz, jnz*

Controlled *call* corruption

Corrupt code in IPE to crash the application

Overwrite secret data with known values

Insert a universal read gadget

Controlled *call* corruption

Corrupt code in IPE to crash the application

Overwrite secret data with known values

Insert a universal read gadget

Overwrite the stored IPE configuration to remove the protection

Live demo!

I’m going to ask Daniel Gruss for his favorite number

TI Embedded Security Portfolio –

Security is hard, TI makes it easier

PSIRT Notification

MSP430FR5xxx and MSP430FR6xxx IP Encapsulation Write Vulnerability

TEXAS INSTRUMENTS

Summary

The IP Encapsulation feature of the Memory Protection Unit may not properly prevent writes to an IPE protected region under certain conditions. This vulnerability assumes an attacker has control of the device outside of the IPE protected region (access to non-protect memory, RAM, and CPU registers).

Vulnerability

TI PSIRT ID

TI-PSIRT-2023-040180

CVE ID

Not applicable.

CVSS Base Score

7.1

CVSS Vector

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Affected Products

- MSP430FR58xx family devices
- MSP430FR59xx family devices
- MSP430FR6xxx family devices

PSIRT Notification

MSP430FR5xxx and MSP430FR6xxx IP Encapsulation Write Vulnerability

Texas Instruments

Summary

The IP Encapsulation feature of the Memory Protection Unit may not properly prevent writes to an IPE protected region under certain conditions. This vulnerability assumes an attacker has control of the device outside of the IPE protected region (access to non-protect memory, RAM, and CPU registers).

Vulnerability

TI PSIRT ID

TI-PSIRT-2023-040180

CVE ID

Not applicable.

CVSS Base Score

7.1

CVSS Vector

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Affected Products

- MSP430FR58xx family devices
- MSP430FR59xx family devices
- MSP430FR6xxx family devices

CVE-2017-5754 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Description

Systems with microprocessors utilizing speculative execution and indirect branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis of the data cache.

Metrics

CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NIST: NVD **Base Score:** 5.6 MEDIUM **Vector:** CVSS:3.0/AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:N/A:N

PSIRT Notification

MSP430FR5xxx and MSP430FR6xxx IP Encapsulation Write Vulnerability

Texas Instruments

Summary

The IP Encapsulation feature of the Memory Protection Unit may not properly prevent writes to an IPE protected region under certain conditions. This vulnerability assumes an attacker has control of the device outside of the IPE protected region (access to non-protect memory, RAM, and CPU registers).

Vulnerability

TI PSIRT ID

TI-PSIRT-2023-040180

CVE ID

Not applicable.

CVSS Base Score

7.1

CVSS Vector

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Affected Products

- MSP430FR58xx family devices
- MSP430FR59xx family devices
- MSP430FR6xxx family devices

CVE-2017-5754 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Description

Systems with microprocessors utilizing speculative execution and indirect branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis of the data cache.

Metrics

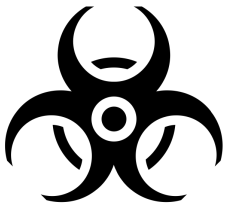
CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NIST: NVD **Base Score:** 5.6 MEDIUM **Vector:** CVSS:3.0/AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:N/A:N

Systematization: IPE attack primitives



	Attack primitive	C ✗	I ✗	Section
Architectural	Controlled `call` corruption *(new)*	◐	●	§3.1
	Code gadget reuse [35]	◐	◐	§3.2
	Interrupt register state [73]	●	●	§3.3
	Interface sanitization [69]	◐	◐	§6.1
Side channels	Cache timing side channel [23, 39]	◐	○	§3.4.1
	Interrupt latency side channel [71]	◐	○	§3.4.2
	Controlled channel [25, 77]	◐	○	§3.4.3
	Voltage fault injection [31, 40]	○	○	§A.1
	DMA contention side channel [7, 8]	○	○	§A.2

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●).
Tested on multiple different MSP430 CPUs.

Systematization: IPE attack primitives

Software-based

	Attack primitive	C ✗	I ✗	Section
Architectural	Controlled `call` corruption *(new)*	◐	●	§3.1
	Code gadget reuse [35]	◐	◐	§3.2
	Interrupt register state [73]	●	●	§3.3
	Interface sanitization [69]	◐	◐	§6.1
Side channels	Cache timing side channel [23, 39]	◐	○	§3.4.1
	Interrupt latency side channel [71]	◐	○	§3.4.2
	Controlled channel [25, 77]	◐	○	§3.4.3
	Voltage fault injection [31, 40]	○	○	§A.1
	DMA contention side channel [7, 8]	○	○	§A.2

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●).
Tested on multiple different MSP430 CPUs.

Systematization: IPE attack primitives

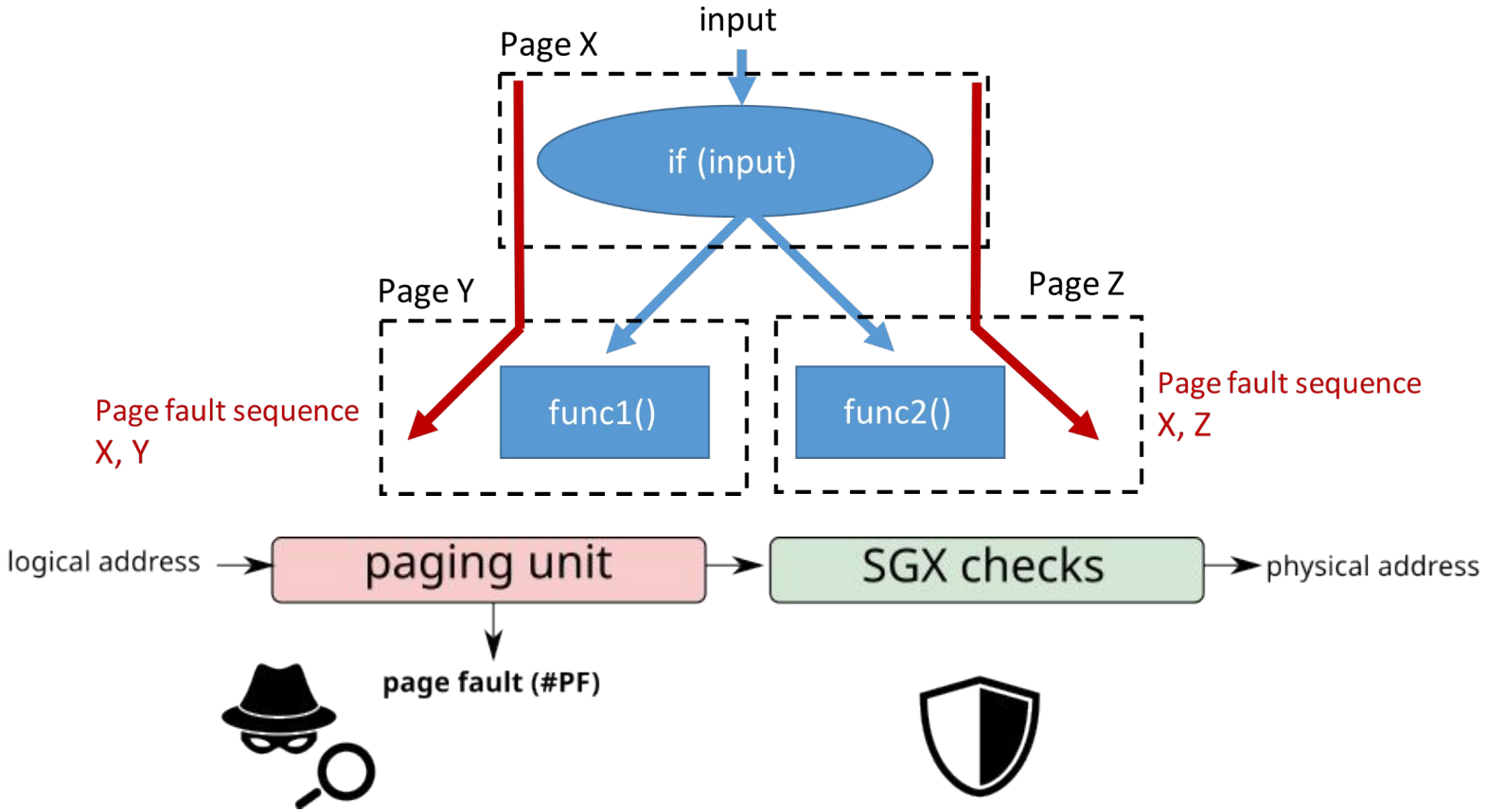
Software-based

	Attack primitive	C ✗	I ✗	Section
Architectural	Controlled `call` corruption *(new)*	◐	●	§3.1
	Code gadget reuse [35]	◐	◐	§3.2
	Interrupt register state [73]	●	●	§3.3
	Interface sanitization [69]	◐	◐	§6.1
Side channels	Cache timing side channel [23, 39]	◐	○	§3.4.1
	Interrupt latency side channel [71]	◐	○	§3.4.2
	Controlled channel [25, 77]	◐	○	§3.4.3
	Voltage fault injection [31, 40]	○	○	§A.1
	DMA contention side channel [7, 8]	○	○	§A.2

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●).
Tested on multiple different MSP430 CPUs.

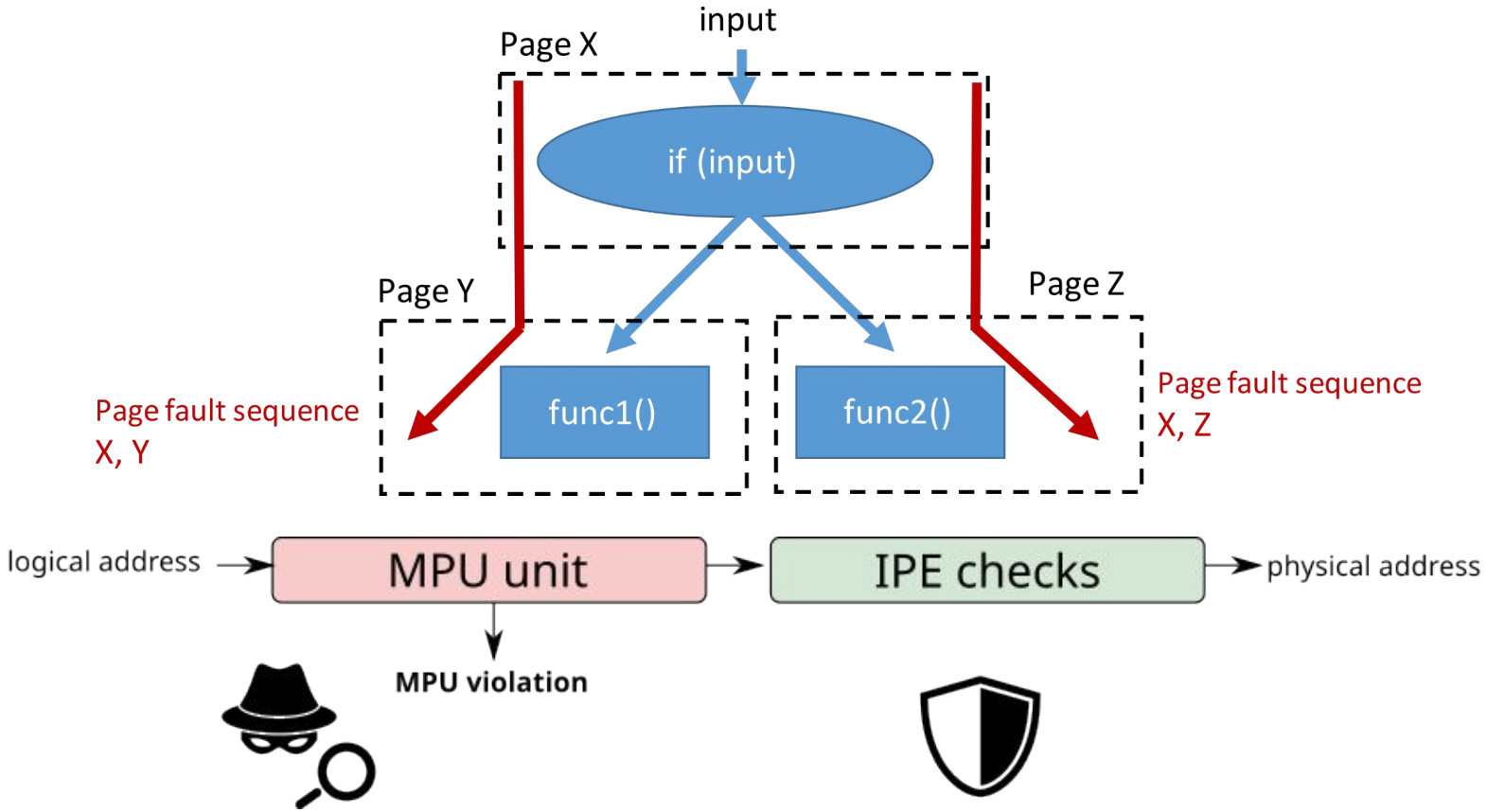

SIDE-CHANNEL LEAKAGE
LIGHTYEAR
SIDE-CHANNEL LEAKAGE EVERYWHERE
imgflip.com

Generalizing “controlled-channel” attacks

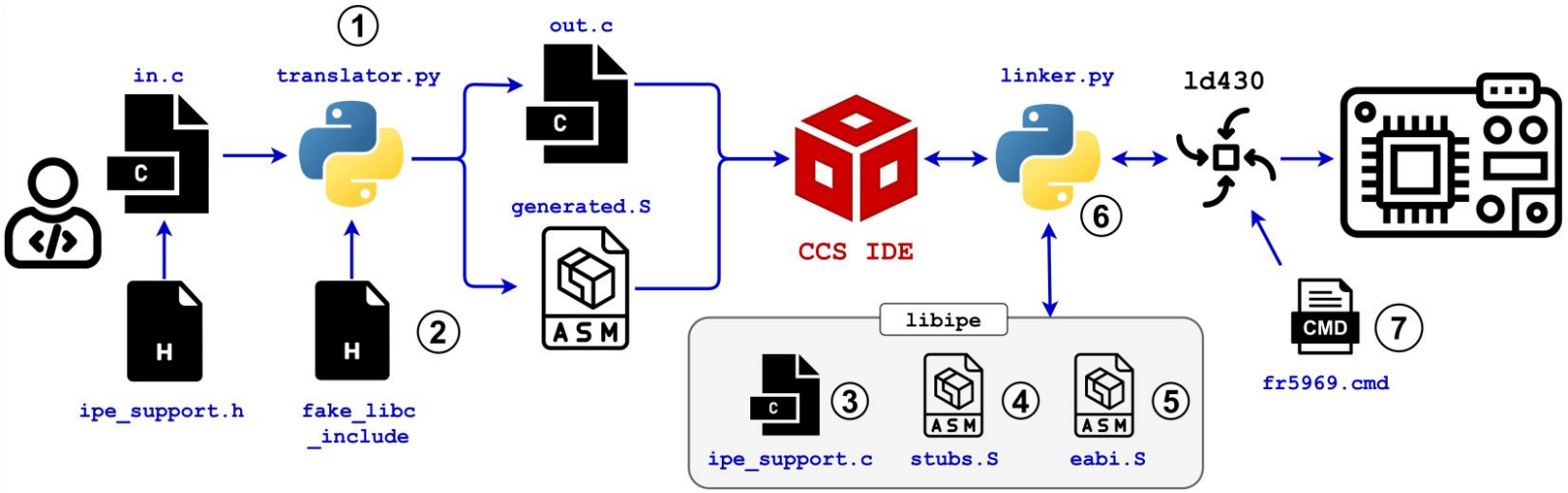


Xu et al.: “Controlled-channel attacks: Deterministic side channels for untrusted operating systems”, Oakland 2015.

Generalizing “controlled-channel” attacks



Software mitigation: MPU to the rescue



- Re-purpose MPU to prevent architectural leakage
- Weaker attacker model → trust reset handler + JTAG

Part III: Extensible memory isolation

Research trends

- **TI MSP430** difficult to do research on:
  - Closed-source hardware and firmware
  - No white-box simulator

	name	year	venue
TI MSP430	IPE [39] 🐞	2014	–
	↳ SIA [63]	2019	HOST
	↳ SICP [64]	2020	JHSS
	↳ Optimized SICP [65]	2022	TECS
	↳ IPE Exposure [20] 🐞	2024	USENIX
	PISTIS [66]	2022	USENIX
	↳ FLAShadow [67]	2024	TIOT
	openIPE *(this work)*	2025	EuroS&P

Research trends

- **TI MSP430** difficult to do research on:
  - Closed-source hardware and firmware
  - No white-box simulator

- **openMSP430:** popular in research
  - Many systems (re-)implement isolation features
  - No compatibility with each other or industry standards
  - Limited applicability to real-world devices

	name	year	venue
openMSP430	SMART [3] 🐞	2012	NDSS
	↪ ERASMUS [51]	2018	DATE
	Sancus 1.0 [52]	2013	USENIX
	↪ Soteria [53]	2015	ACSAC
	↪ Towards Availability [11]	2016	MASS
	↪ Sancus 2.0 [2] 🐞	2017	TOPS
	↪ Sancus$_V$ [33] 🐞	2020	CSF
	↪ Aion [8]	2021	CCS
	↪ Authentic Execution [54]	2023	TOPS
	de Clercq et al. [7]	2014	ASAP
	VRASED [4] 🐞	2019	USENIX
	↪ APEX [50] 🐞	2020	USENIX
	↪ ASAP [55]	2022	DAC
	↪ RARES [56]	2023	arXiv
	↪ RATA [57]	2021	CCS
	↪ CASU [58]	2022	ICCAD
	↪ VERSA [59]	2022	S&P
	↪ ACFA [60]	2023	USENIX
	GAROTA [61]	2022	USENIX
	IDA [10]	2024	NDSS
	UCCA [62]	2024	TCAD
TI MSP430	IPE [39] 🐞	2014	–
	↪ SIA [63]	2019	HOST
	↪ SICP [64]	2020	JHSS
	↪ Optimized SICP [65]	2022	TECS
	↪ IPE Exposure [20] 🐞	2024	USENIX
	PISTIS [66]	2022	USENIX
	↪ FLAShadow [67]	2024	TIOT
	openIPE *(this work)*	2025	EuroS&P

Overlapping vulnerabilities

Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic

Jo Van Bulck
imec-DistriNet, KU Leuven
jo.vanbulck@cs.kuleuven.be

Frank Piessens
imec-DistriNet, KU Leuven
frank.piessens@cs.kuleuven.be

Raoul Strackx
imec-DistriNet, KU Leuven
raoul.strackx@cs.kuleuven.be

Mind the Gap: Studying the Insecurity of Provably Secure Embedded Trusted Execution Architectures

Marton Bognar
marton.bognar@kuleuven.be
imec-DistriNet, KU Leuven
3001 Leuven, Belgium

Jo Van Bulck
jo.vanbulck@kuleuven.be
imec-DistriNet, KU Leuven
3001 Leuven, Belgium

Frank Piessens
frank.piessens@kuleuven.be
imec-DistriNet, KU Leuven
3001 Leuven, Belgium

A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes

Jo Van Bulck
imec-DistriNet, KU Leuven
jo.vanbulck@cs.kuleuven.be

David Oswald
The University of Birmingham, UK
d.f.oswald@cs.bham.ac.uk

Eduard Marin
The University of Birmingham, UK
e.marin@cs.bham.ac.uk

Abdulla Aldoseri
The University of Birmingham, UK
axa1170@student.bham.ac.uk

Flavio D. Garcia
The University of Birmingham, UK
f.garcia@cs.bham.ac.uk

Frank Piessens
imec-DistriNet, KU Leuven
frank.piessens@cs.kuleuven.be

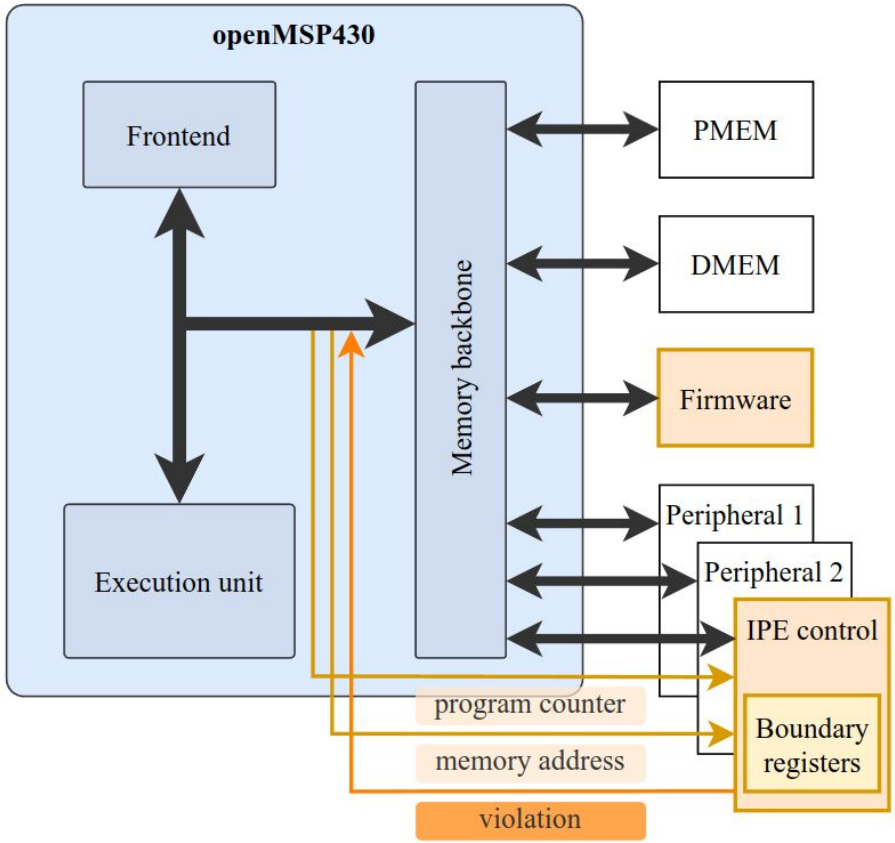
Intellectual Property Exposure: Subverting and Securing Intellectual Property Encapsulation in Texas Instruments Microcontrollers

Marton Bognar, Cas Magnus, Frank Piessens, Jo Van Bulck

DistriNet, KU Leuven, 3001 Leuven, Belgium

Our proposal: openIPE

- **Flexible isolation** primitive
  - Based on the IPE specification
  - With protected firmware
  - But freely configurable!
- Includes proposed **hardware fixes** for IPE



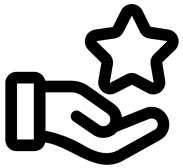
Our proposal: openIPE

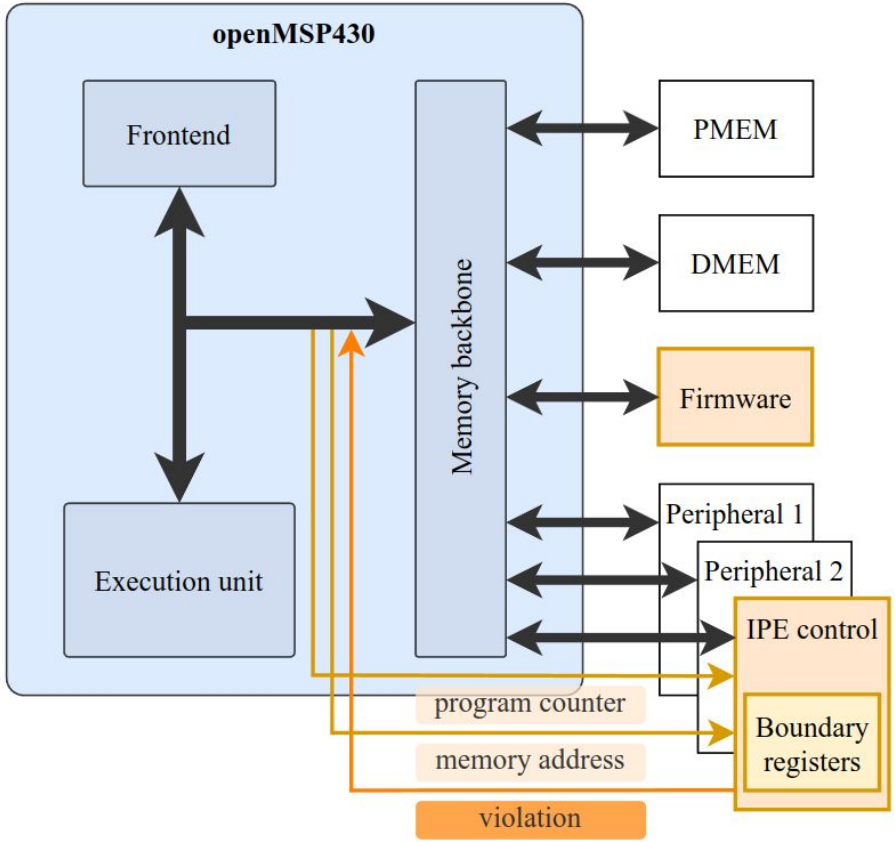
- **Flexible isolation** primitive
  - Based on the IPE specification
  - With protected firmware
  - But freely configurable!
- Includes proposed **hardware fixes** for IPE

KU LEUVEN **Embedded Security Portfolio –**

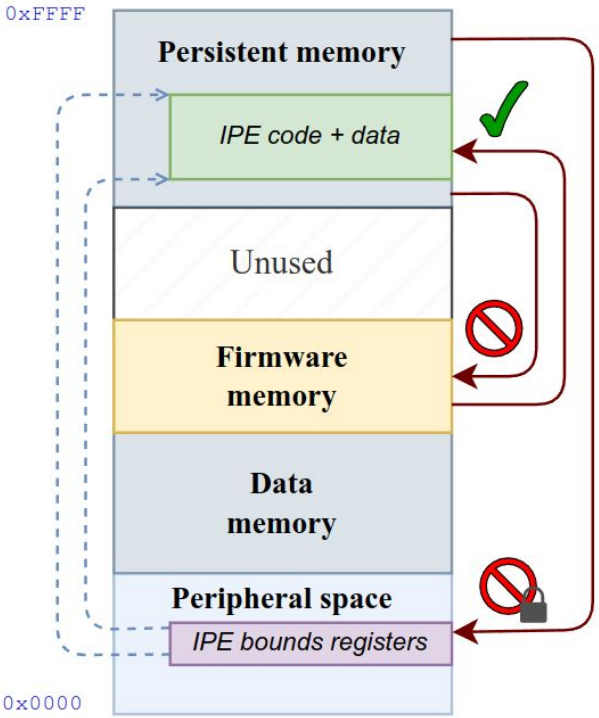
Security is hard, ~~TI~~ makes it easier

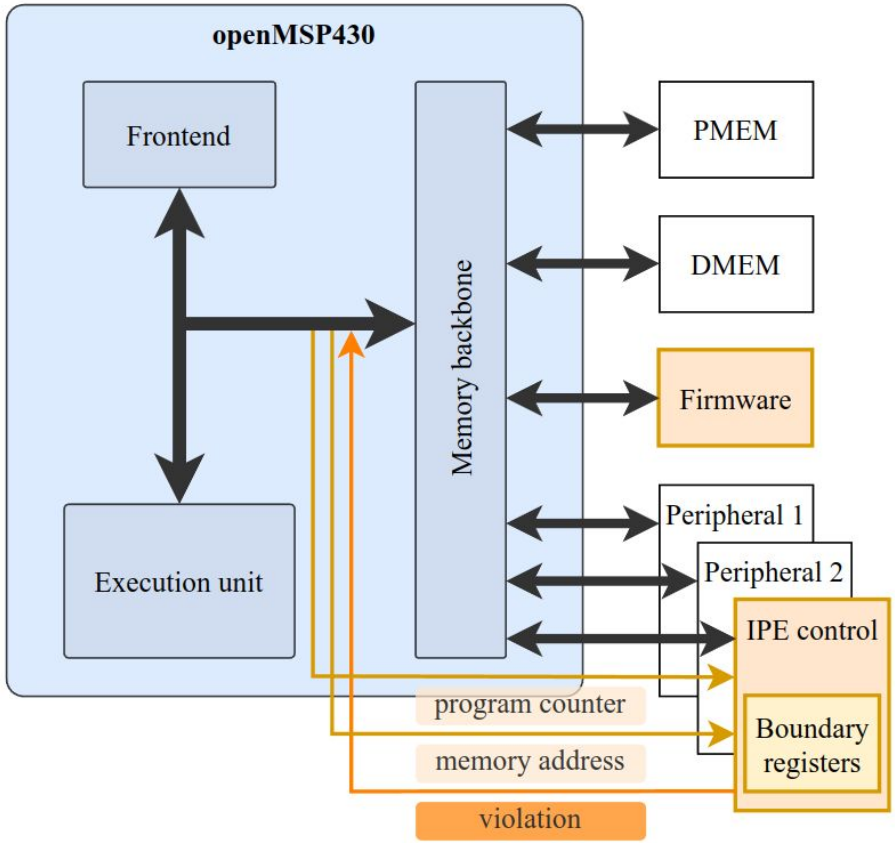
openIPE





Our proposal: openIPE





Case study: Secure interrupt handling

Approach	Secure scheduling	Architectural protection	Interrupt latency mitigation	Untrusted interrupts
Software disable	○	◐	●	○
Hardware disable	○	●	●	○
`SW-IRQ` (de Clercq, 2014)	◐	●	○	●
`FW-IRQ` *(our proposal)*	◐	●	●	●

Design	LUTs	FFs	Δ Software
openIPE (baseline)	2,582	1,191	–
Software disable	–	–	8 bytes / 6 cycles
Hardware disable	2,581 (-1)	1,191	–
`SW-IRQ`	2,597 (+15)	1,191	282 bytes / 132 cycles
`FW-IRQ`	2,577 (-5)	1,190 (-1)	674 bytes / 345 cycles

The ~~proof~~ validation is in the pudding

Hardware security validation: Unit tests

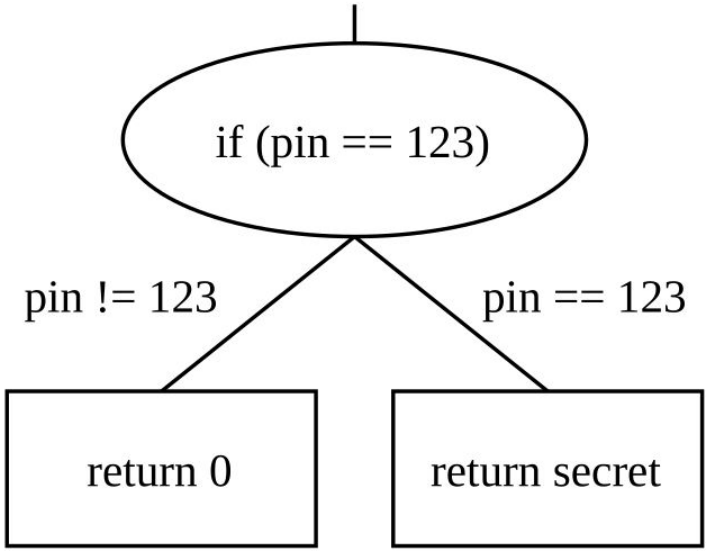
- Functional and security tests
- Backwards compatibility for (future) extensions

# tests	Tested functionality
4	IPE boundary setup
2	Modification of boundary registers
3	Protection from untrusted code
3	Protection from the debugger
2	Protection from DMA
1	Normal access from inside the IPE region
4	Protection from known attacks
4	Protection of the firmware region
3	Case study behavior
62	openMSP430 regression tests

Software security validation: Symbolic execution

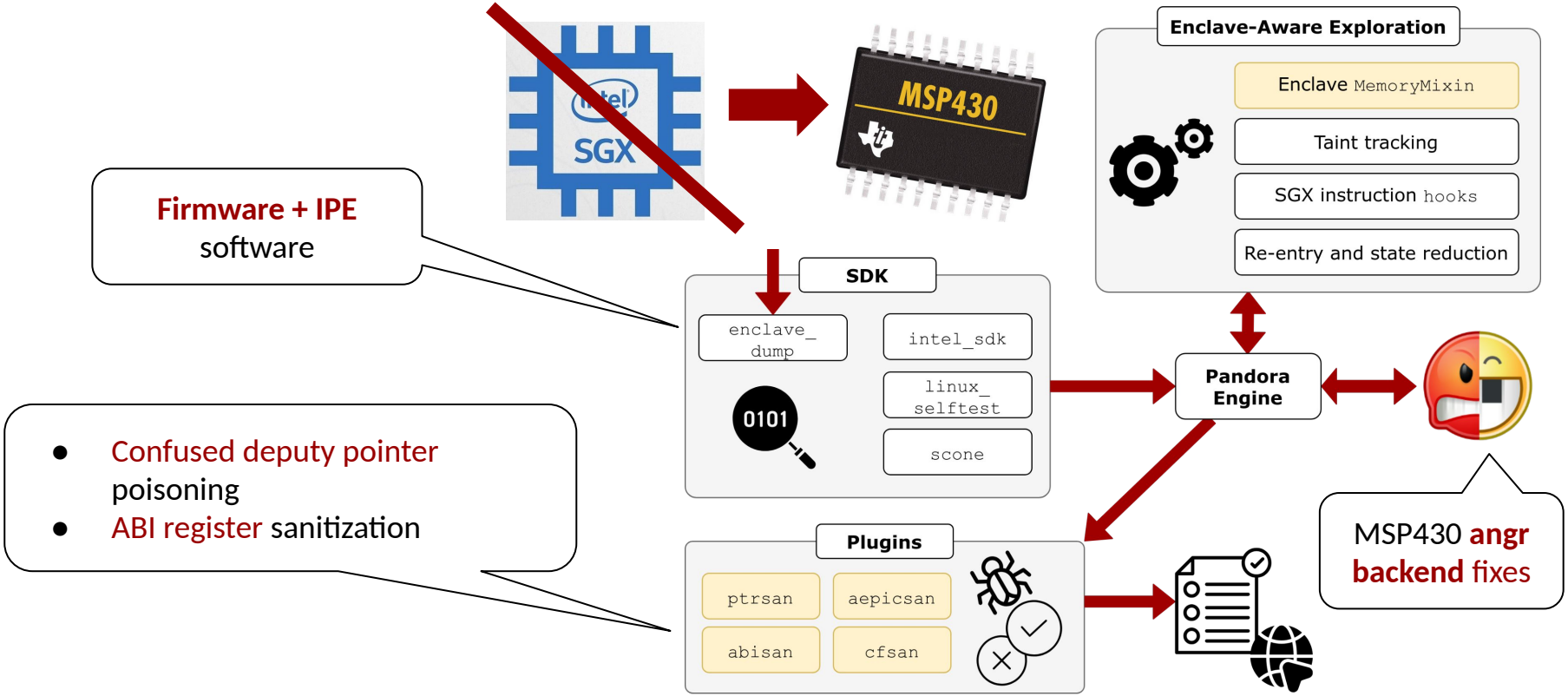
```
1 int ecall(int pin){
2     if(pin == 123){
3         return secret;
4     } else {
5         return 0;
6     }
7 }
```

https://angr.io/



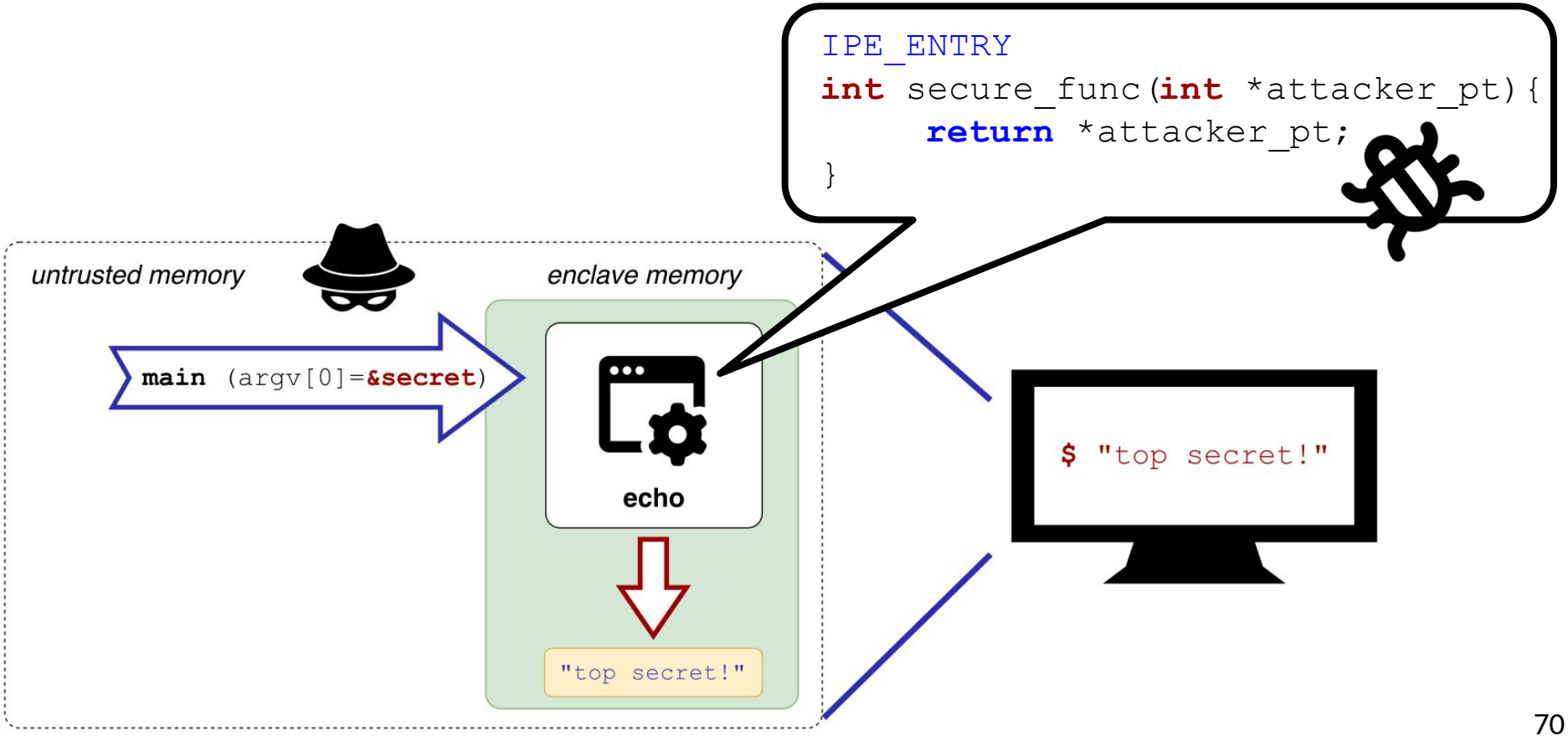
- Symbolic execution uses a constraint solver
- Execution works on instruction-level, i.e., as close to the binary as possible

Principled symbolic ~~Intel SGX~~ TI IPE enclave validation



Alder et al. “Pandora: Principled Symbolic Validation of Intel SGX Enclave Runtimes”, S&P 2024.

Example: Insufficient pointer validation



Report PointerSanitizationPlugin

Plugin description: Validates attacker-tainted pointer dereferences.

Analyzed 'ipe-hello.elf', with 'openIPE' enclave runtime. Ran for 0:00:01.850551 on 2025-02-20_14-25-42.

Enclave info: Address range is [(0x8000, 0xe3df)]

Summary: Found 2 unique WARNING issues; 2 unique CRITICAL issues.

Report summary

Severity	Reported issues
WARNING	• *Attacker tainted read inside enclave* at 0x802a • *Attacker tainted read inside enclave* at 0x8022
CRITICAL	• *Non-tainted read outside enclave* at 0x5c98 • *Unconstrained read* at 0x81c4

Issues reported at 0x81c4 2 ipe_func_internal CRITICAL Unconstrained read

Unconstrained read CRITICAL IP=0x81c4

Plugin extra info

Key	Value
Address	<BV16 r15_attacker_15_16>
Attacker tainted	True
Length	2
Pointer range	[0x0, 0xffff]
Pointer can wrap address space	True
Pointer can lie in enclave	True
Extra info	Read address may lie inside or outside enclave

Execution state info

Disassembly

```
000081b4 <ipe_func_internal>:
    81b4:       04 12           push    r4
    81b6:       04 41           mov     r1,     r4
    81b8:       24 53           incd    r4
    81ba:       21 83           decd    r1
    81bc:       84 4f fc ff     mov     r15,    -4(r4)  ;0xfffc(r4)
    81c0:       1f 44 fc ff     mov     -4(r4), r15     ;0xfffc(r4)
    81c4:       2f 4f           mov     @r15,   r15
    81c6:       21 53           incd    r1
    81c8:       34 41           pop     r4
    81ca:       30 41           ret
```

Issues reported at 0x81c4 2 ipe_func_internal CRITICAL Unconstrained read

Unconstrained read CRITICAL IP=0x81c4

Plugin extra info

Key	Value
Address	<BV16 r15_attacker_15_16>
Attacker tainted	True
Length	2
Pointer range	[0x0, 0xffff]
Pointer can wrap address space	True
Pointer can lie in enclave	True
Extra info	Read address may lie inside or outside enclave

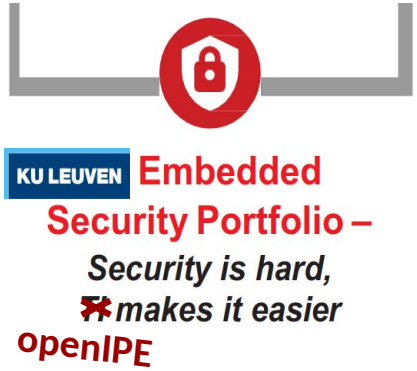
Execution state info

Disassembly

```
000081b4 <ipe_func_internal>:
    81b4:       04 12           push    r4
    81b6:       04 41           mov     r1,     r4
    81b8:       24 53           incd    r4
    81ba:       21 83           decd    r1
    81bc:       84 4f fc ff     mov     r15,    -4(r4)  ;0xfffc
    81c0:       1f 44 fc ff     mov     -4(r4), r15
    81c4:       2f 4f           mov     @r15,   r15
    81c6:       21 53           incd    r1
    81c8:       34 41           pop     r4
    81ca:       30 41           ret
```

Conclusions and outlook

- **IPE Exposure:** First security analysis of Texas Instruments IPE
  - Novel vulnerability: *controlled call corruption*
  - Reproduction of other known primitives, including side channels
  - *Complete leakage of protected code and data*
  - Software-only mitigation via MPU
- **openIPE:** Open-source extensible memory isolation
  - Hardware + firmware + software co-design
  - *Paper to appear at EuroS&P '25*



https://github.com/martonbognar/ipe-exposure

https://github.com/martonbognar/openipe