SAND2020-0912C

µRAI : Securing Embedded Systems with Return Address Integrity

Naif Saleh Almakhdhub
Purdue University and
King Saud University

Abraham A. Clements
Sandia National Labs

Saurabh Bagchi
Purdue University

Mathias Payer
EPFL

Problem

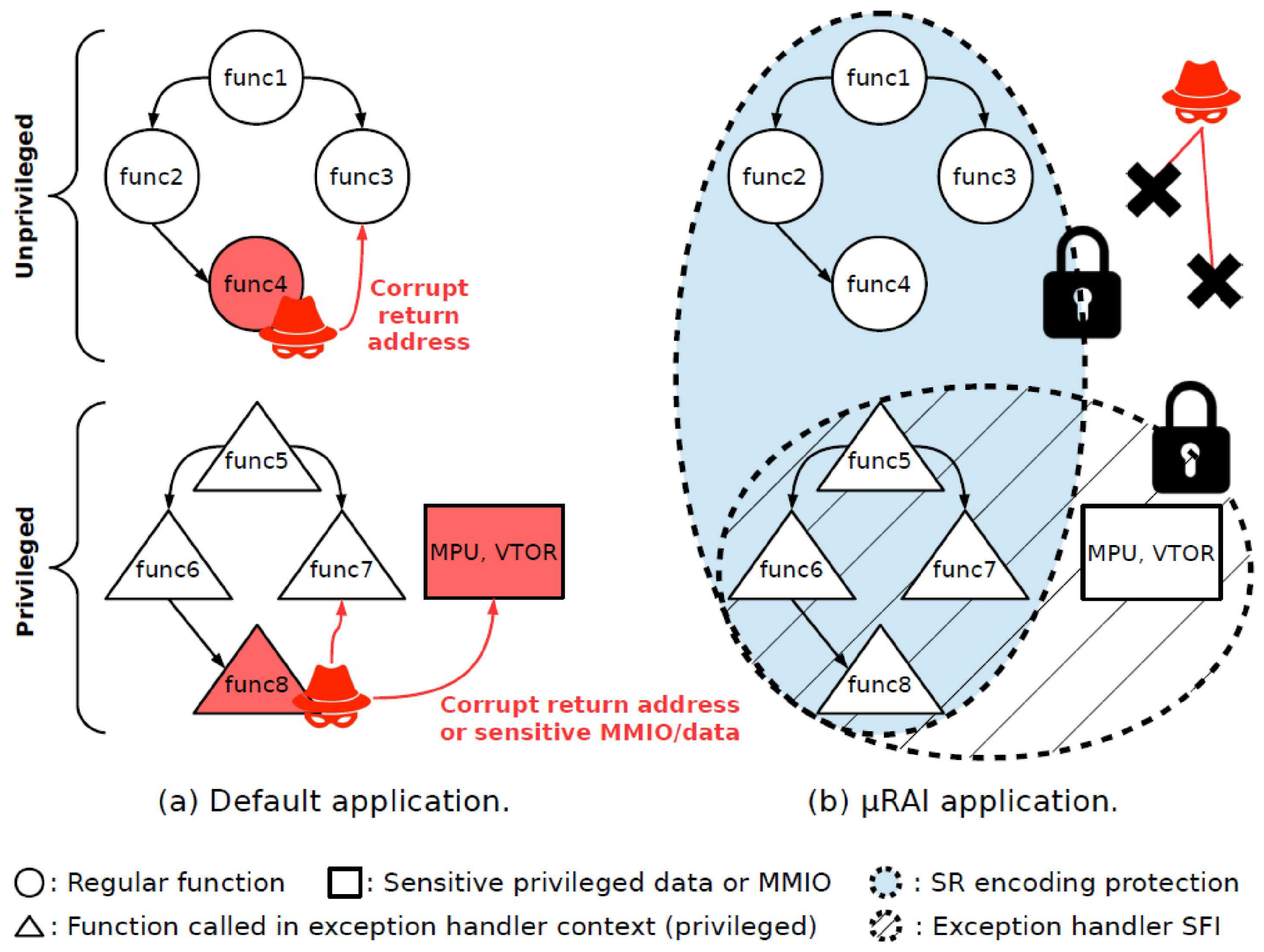
- Microcontroller systems (**MCUS**) are a significant portion of **embedded systems** and **IoT**
- **MCUS** lack basic defenses and are **vulnerable** to **control-flow hijacking attacks** such as Return Oriented Programming (**ROP**)
- Existing defenses either have limited security guarantees, high runtime overhead, or require special hardware features

Objectives

- Prevent ROP style attacks against MCUS by enforcing the Return Address Integrity (**RAI**) Property
- Apply the defense with **low runtime overhead**
- Apply the defense **without** requiring special hardware

µRAI

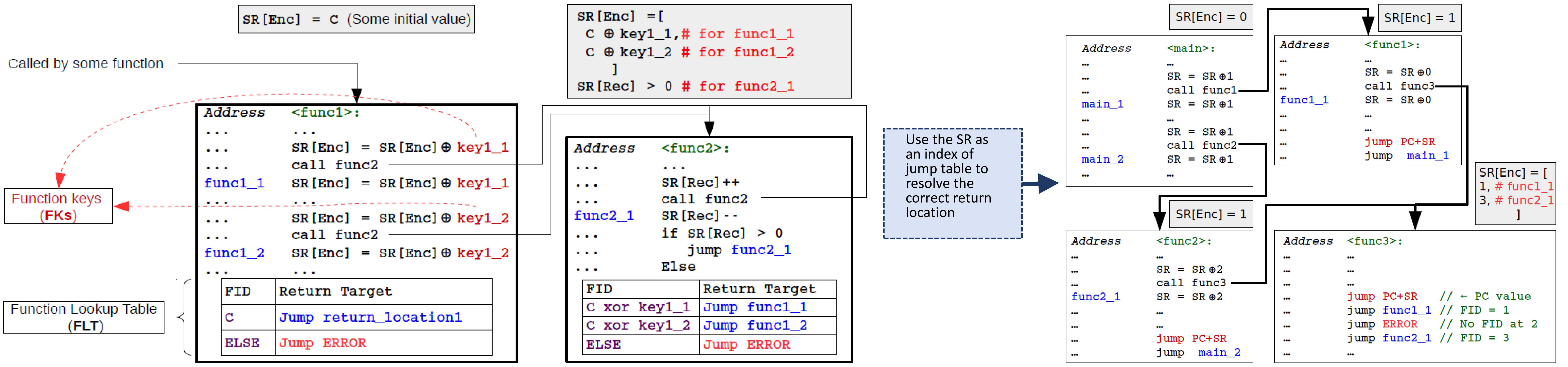
- Analyzes the call graph statically to identify the possible return targets of each function
- Transforms the set of return targets to a jump table and places it in R+X memory
- Encodes a general purpose register called the State Register (**SR**), which is **never spilled** and is exclusively used by µRAI
- Uses the SR at run time to resolve the correct return location from the jump table
- **Enforces the RAI property** since the SR and jump table are **inaccessible** to and adversary
- Enforces Software Fault Isolation (SFI) on functions callable within an exception handler context to protect sensitive Memory Mapped IO (MMIO) such as the MPU
- Partitions the SR into segments to curb path explosion
- Applies a type-based CFI for forward edges



(a) Default application.

(b) µRAI application.

Compiler Transformation



```
SR[Enc]  =  C  (Some initial value)
```

```
Address    <func1>:
...        ...
...        SR[Enc] = SR[Enc]⊕ key1_1
...        call func2
func1_1    SR[Enc] = SR[Enc]⊕ key1_1
...        ...
...        SR[Enc] = SR[Enc]⊕ key1_2
...        call func2
func1_2    SR[Enc] = SR[Enc]⊕ key1_2
...        ...
```

FID	Return Target
C	Jump return_location1
ELSE	Jump ERROR

```
SR[Enc]  =[
 C ⊕ key1_1,# for func1_1
 C ⊕ key1_2 # for func1_2
    ]
SR[Rec] > 0 # for func2_1
```

```
Address    <func2>:
...        ...
...        SR[Rec]++
...        call func2
func2_1    SR[Rec]--
...        if SR[Rec] > 0
...           jump func2_1
...        Else
```

FID	Return Target
C xor key1_1	Jump func1_1
C xor key1_2	Jump func1_2
ELSE	Jump ERROR

Use the SR as an index of jump table to resolve the correct return location

```
SR[Enc] = 0

Address        <main>:
…              …
…              SR = SR ⊕1
…              call func1
main_1         SR = SR ⊕1
…              …
…              SR = SR ⊕1
…              call func2
main_2         SR = SR ⊕1
…              …
```

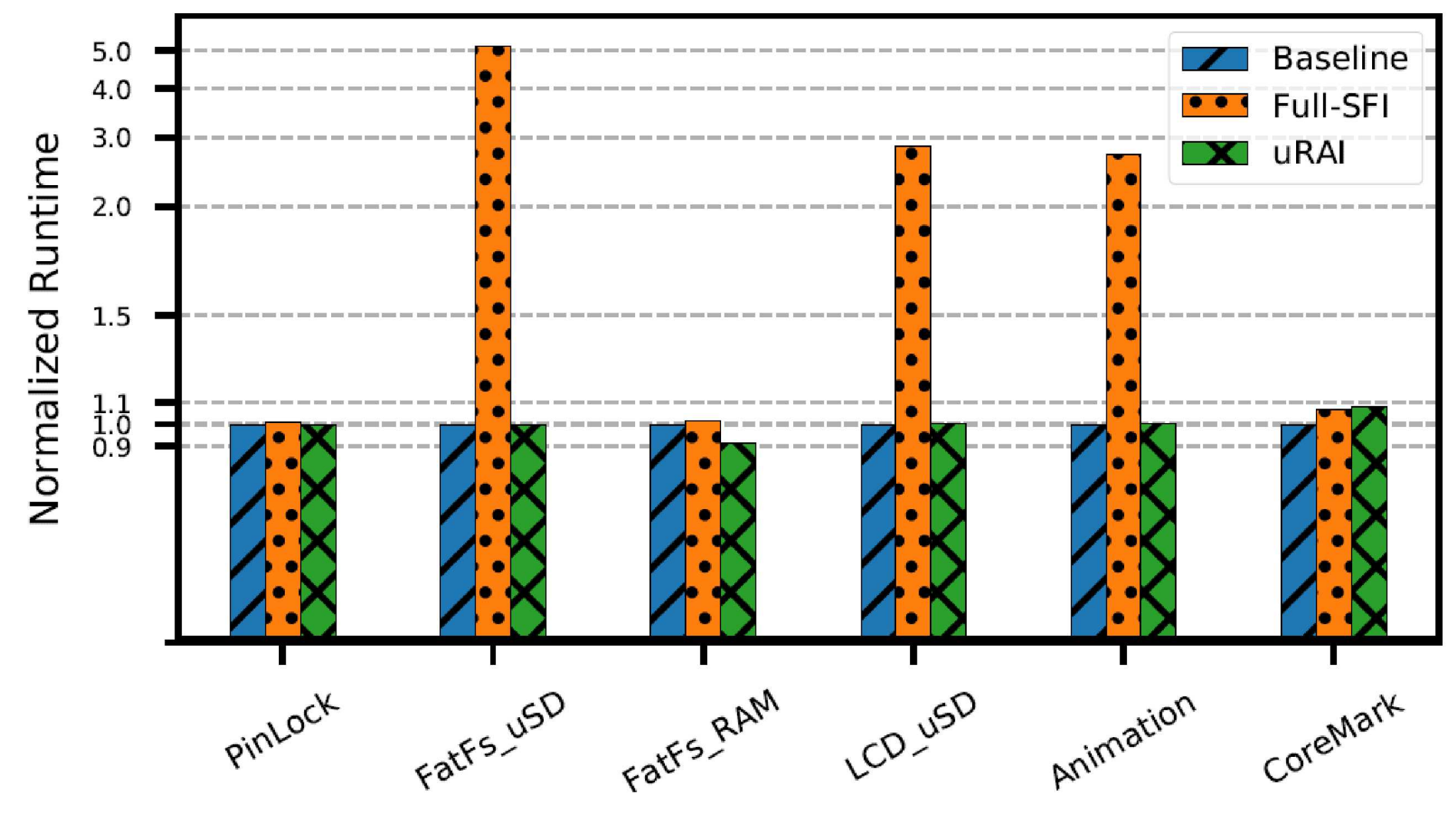
```
SR[Enc] = 1

Address        <func1>:
…              …
…              SR = SR ⊕0
…              call func3
func1_1        SR = SR ⊕0
…              …
…              …
…              jump PC+SR
…              jump  main_1
```

```
SR[Enc] = 1

Address        <func2>:
…              …
…              SR = SR ⊕2
…              call func3
func2_1        SR = SR ⊕2
…              …
…              …
…              jump PC+SR
…              jump  main_2
```

```
SR[Enc] = [
1, # func1_1
3, # func2_1
        ]

Address     <func3>:
…           …
…           …
…           …
…           jump PC+SR    // ← PC value
…           jump func1_1 // FID = 1
…           jump ERROR    // No FID at 2
…           jump func2_1 // FID = 3
…           …
```

Evaluation

Runtime



µRAI enforces the RAI property with low overhead in contrast to mechanisms requiring full-SFI

Comparison to backward edge Type-based CFI

App	Type-based CFI Target Set	
	Max.	Ave.
PinLock	8	3
FatFs_uSD	94	21
FatFs_RAM	94	27
LCD_uSD	49	11
Animation	49	11
CoreMark	52	12

µRAI eliminates the remaining attack surface for control-flow bending attacks

Security

Attack	Prevented
Buffer overflow	✓
Arbitrary write	✓
Stack pivot	✓

µRAI prevents all control-flow hijacking attack scenarios targeting return addresses

References

[1] Naif Saleh Almakhdhub, Abraham A Clements, Saurabh Bagchi, and Mathias Payer.In *The Annual Network and Distributed System Security Symposium (NDSS), 2020*

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA000XXXX SAND20XX-XXXX.