SCIENCE ▪ PASSION ▪ TECHNOLOGY

TU Graz

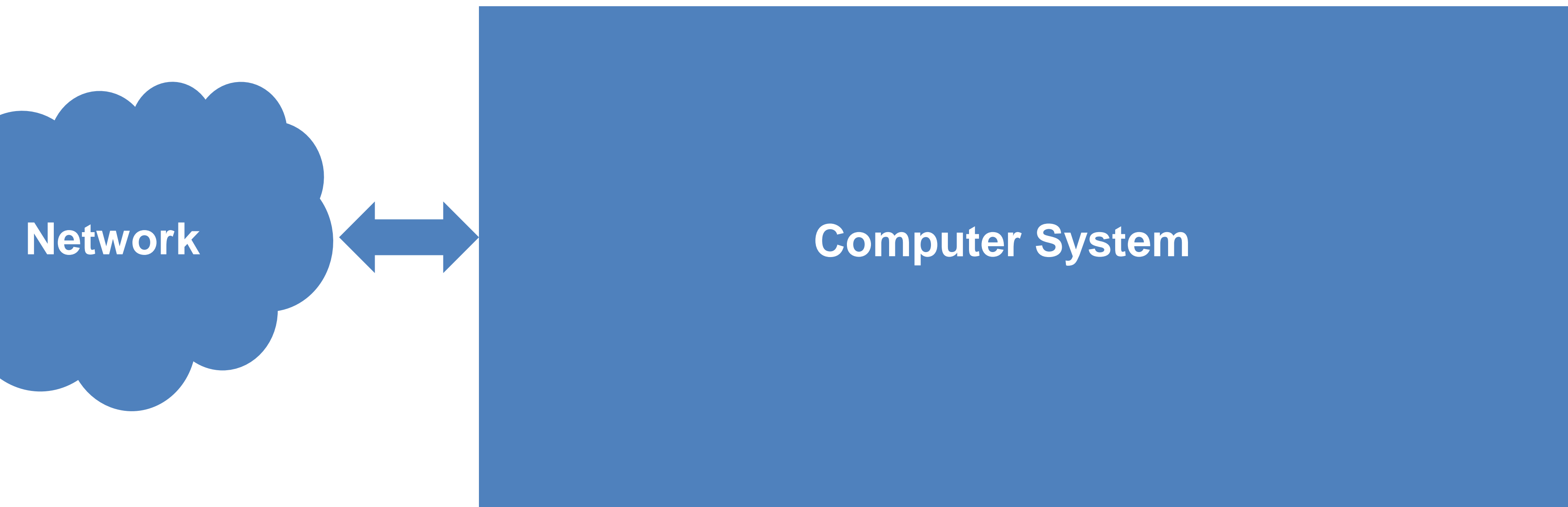
Protecting RISC-V Processors against Physical Attacks

Stefan Mangard, Robert Schilling, Thomas Unterluggauer, Mario Werner

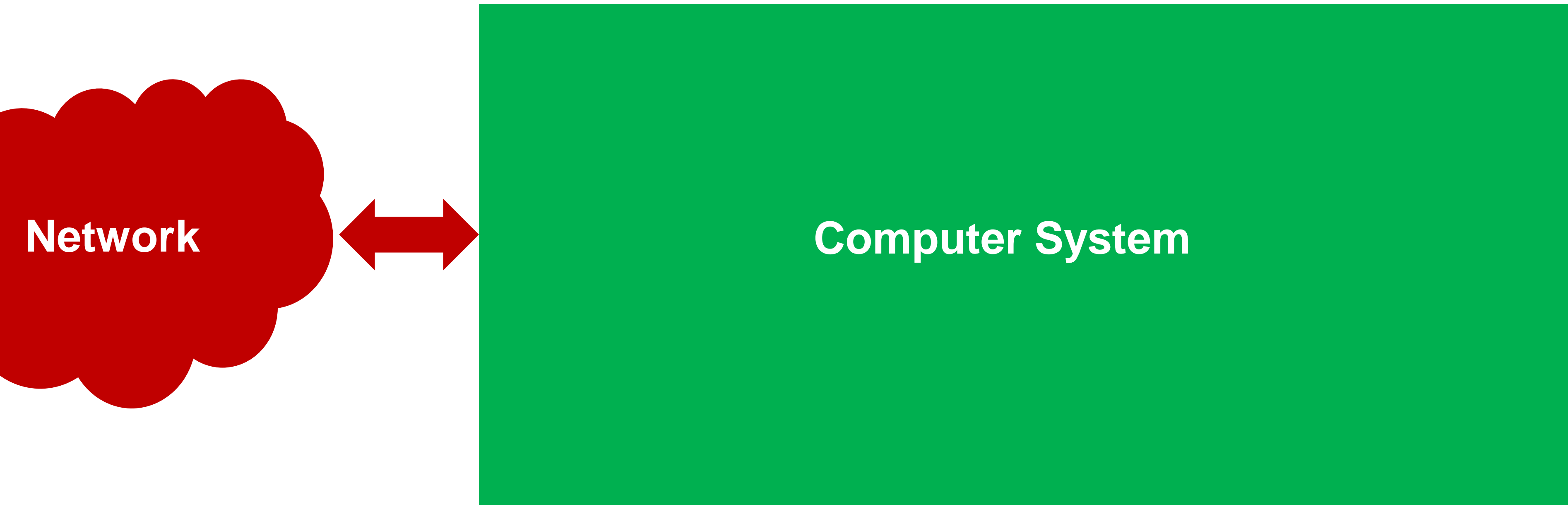
Graz University of Technology

March 28th, 2019

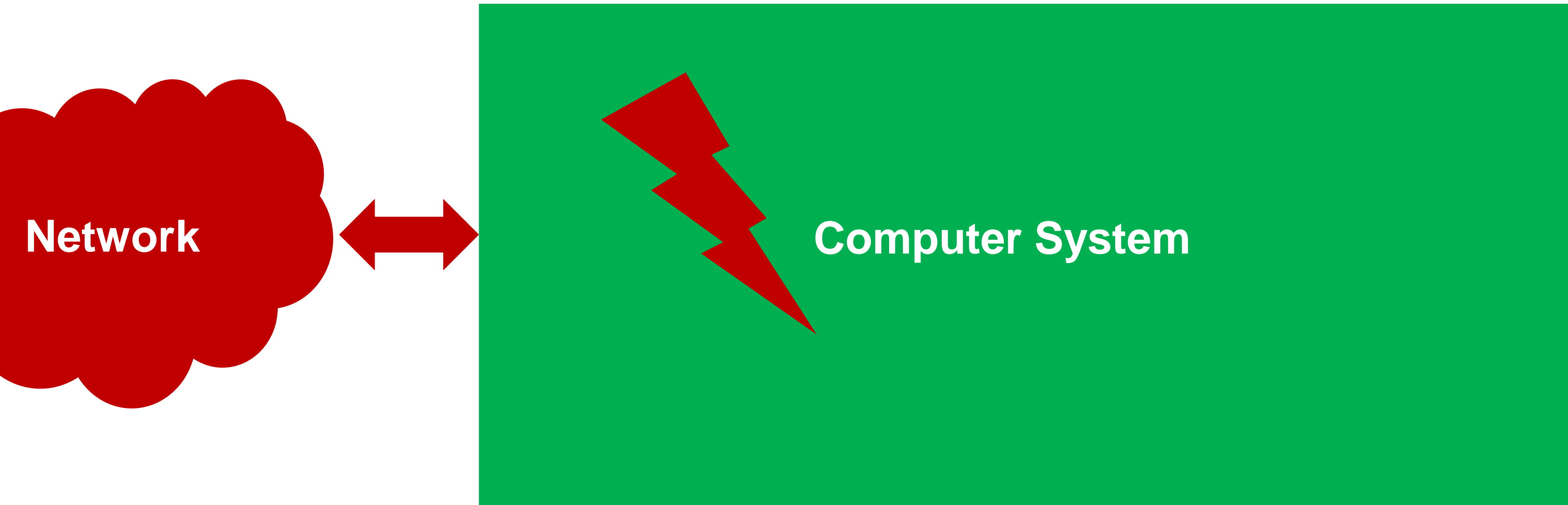
Attack Settings



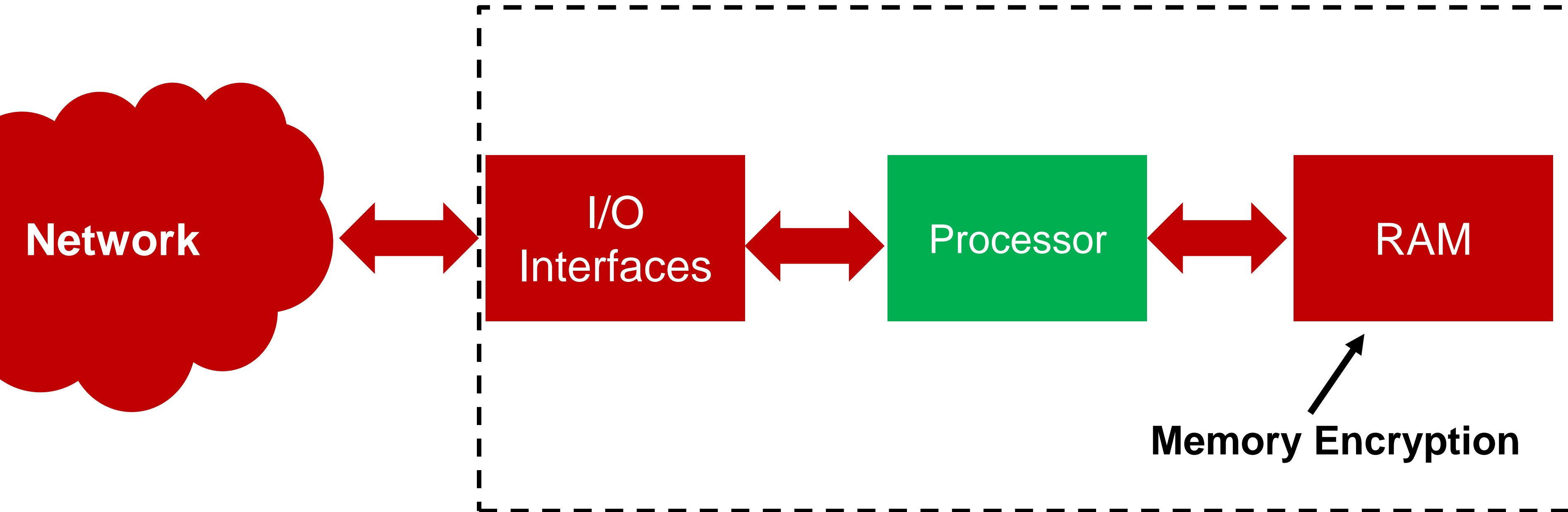
Classic Setting: Attacks Via The (Network) Interfaces



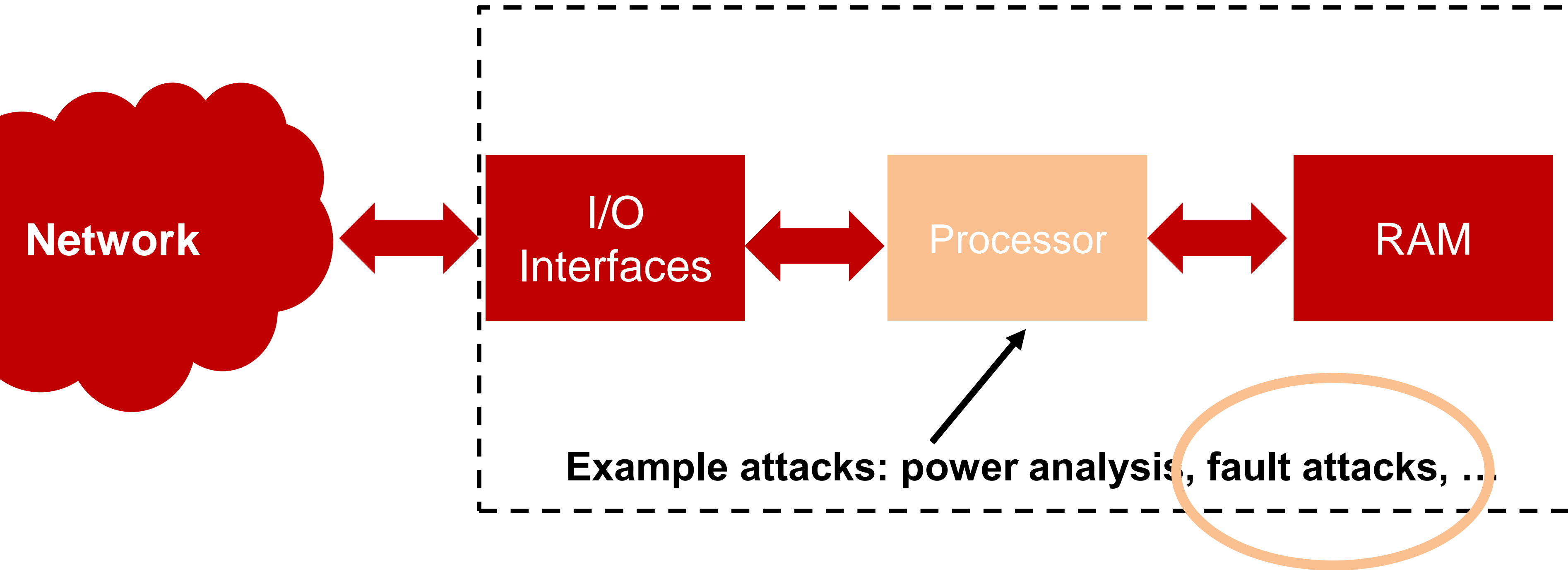
Classic Setting: Attacks Via The (Network) Interfaces



The Secure Processor Setting



Physical Attack Setting: Also Processor is Attacked



Small Faults – Big Effects

1 bit

- determines your privilege level
- determines your access rights
- determines whether a password is considered correct or not
- Changes completely the meaning of an opcode
- …

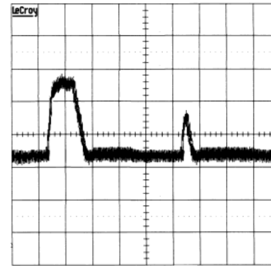
Fault Attacks - Relevance

- Traditional fields
  - secure elements
  - smart cards, …

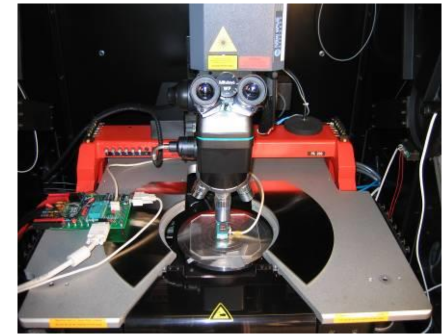
- New fields:
  - IoT, automotive, etc.
  - Example: Privilege escalation on Linux (Niek Timmers, Cristofaro Mune: Escalating Privileges in Linux Using Voltage Fault Injection. FDTC 2017: 1-8)

Fault Attacks – How?

- Most popular techniques
  - Voltage glitches
  - Laser

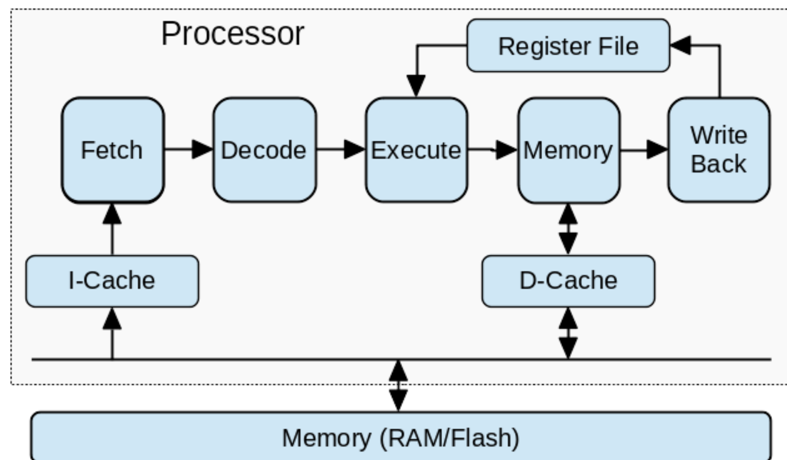


Rowhammer



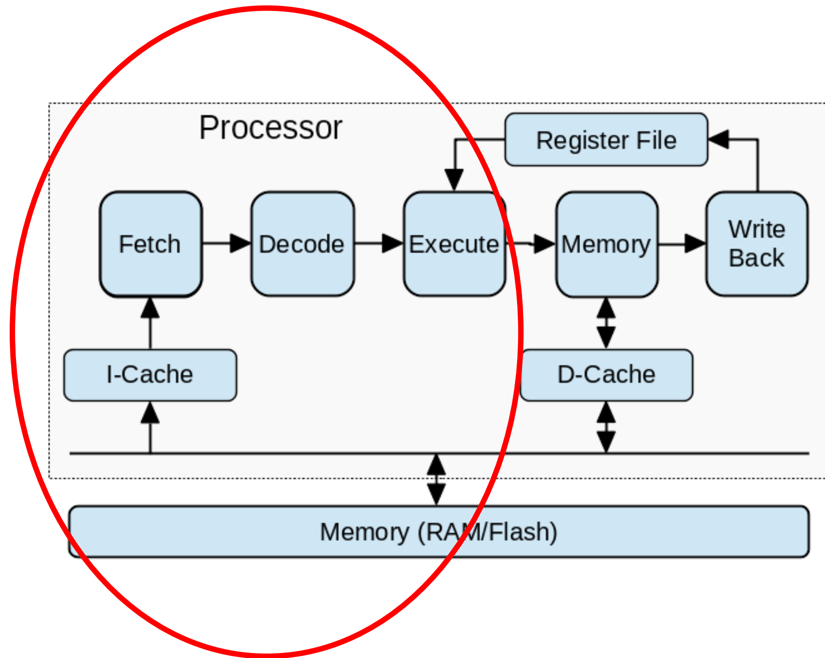
- Effects
  - Changes in data, pointers, …
  - Changes in program flow: skip of instruction pointer, induction of branches, …

Fault Attacks – What Can We Do?



Redundancy

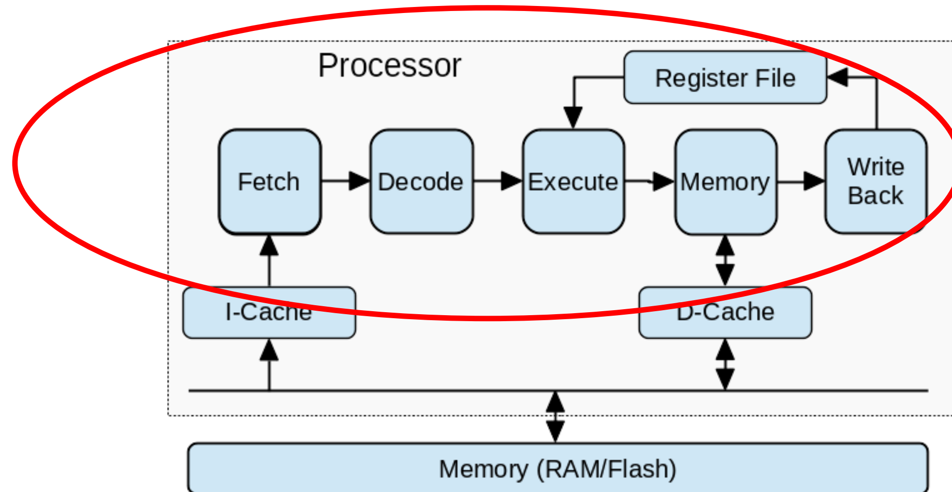
Fault Attacks – What Can We Do?



- **Control flow graph integrity**

Mario Werner, Thomas Unterluggauer, David Schaffenrath, Stefan Mangard. “Sponge-Based Control-Flow Protection for IoT Devices”. In: Euro S&P 2018

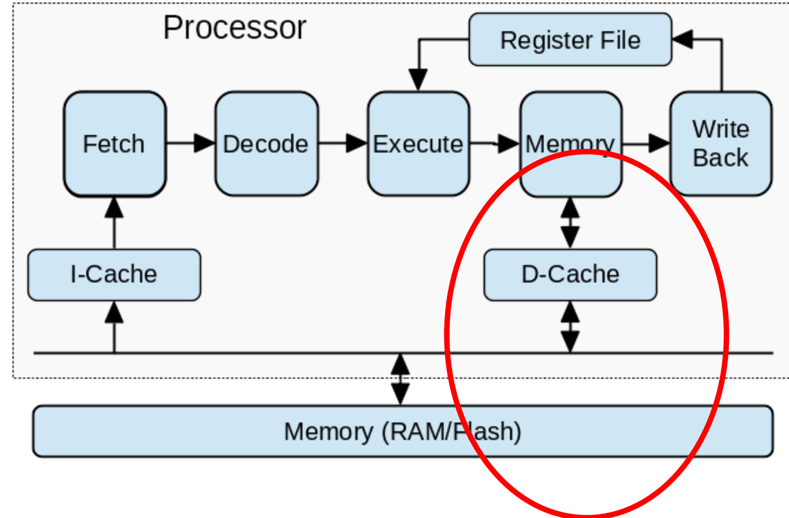
Fault Attacks – What Can We Do?



- **Branch Integrity**

Robert Schilling, Mario Werner, Stefan Mangard. “Securing Conditional Branches in the Presence of Fault Attacks”. In: DATE 2018

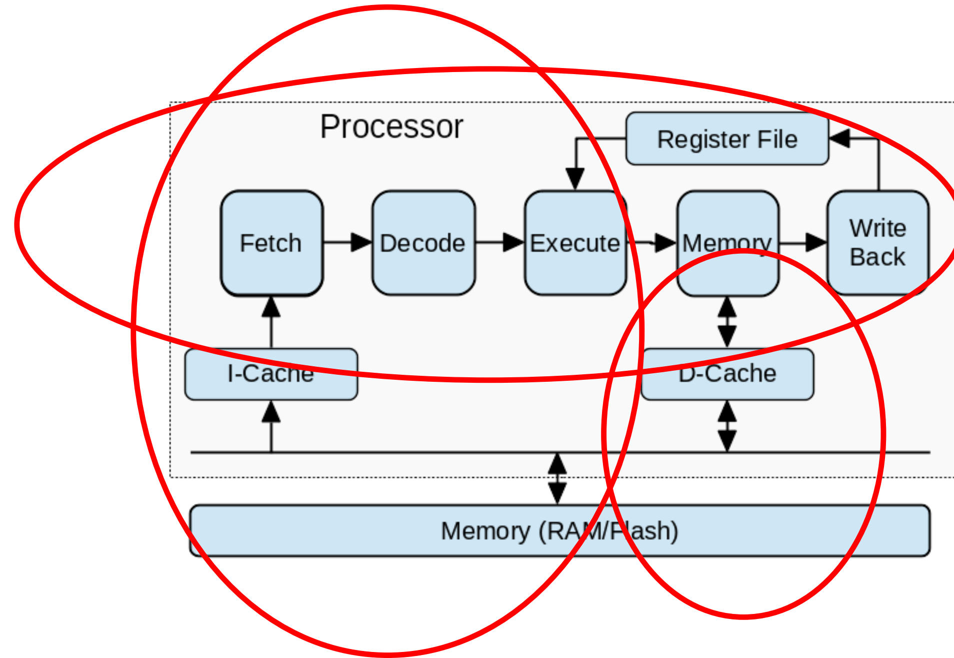
Fault Attacks – What Can We Do?



- **Memory Access**

Robert Schilling, Mario Werner, Pascal Nasahl, Stefan Mangard. “Pointing in the Right Direction-Securing Memory Accesses in a Faulty World”. In: ACSAC 2018

Fault Attacks – What can we do?

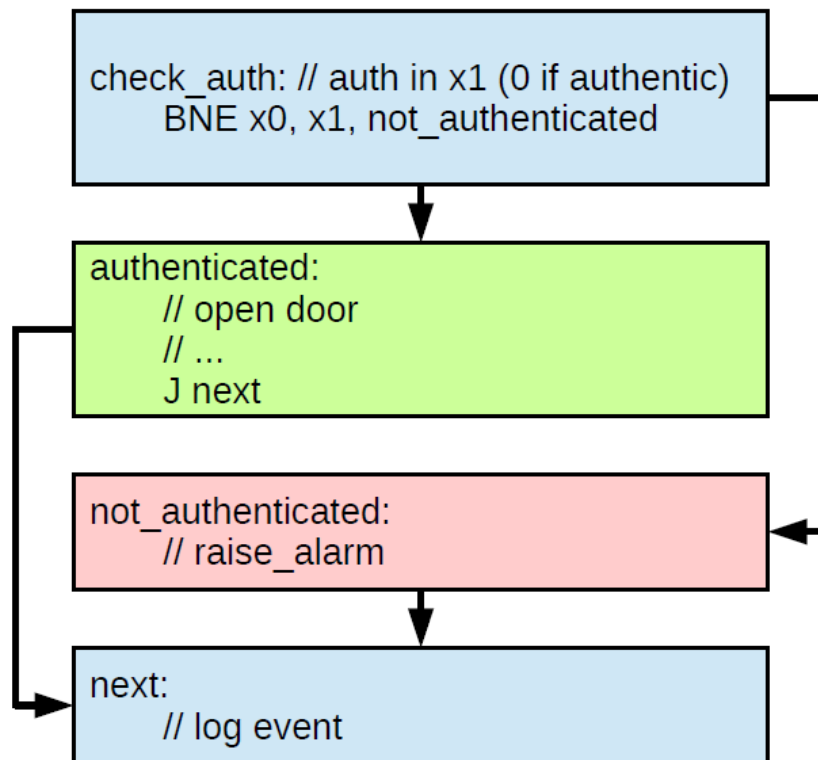


Sponge-Based Control-Flow Protection

Control Flow Integrity is Critical

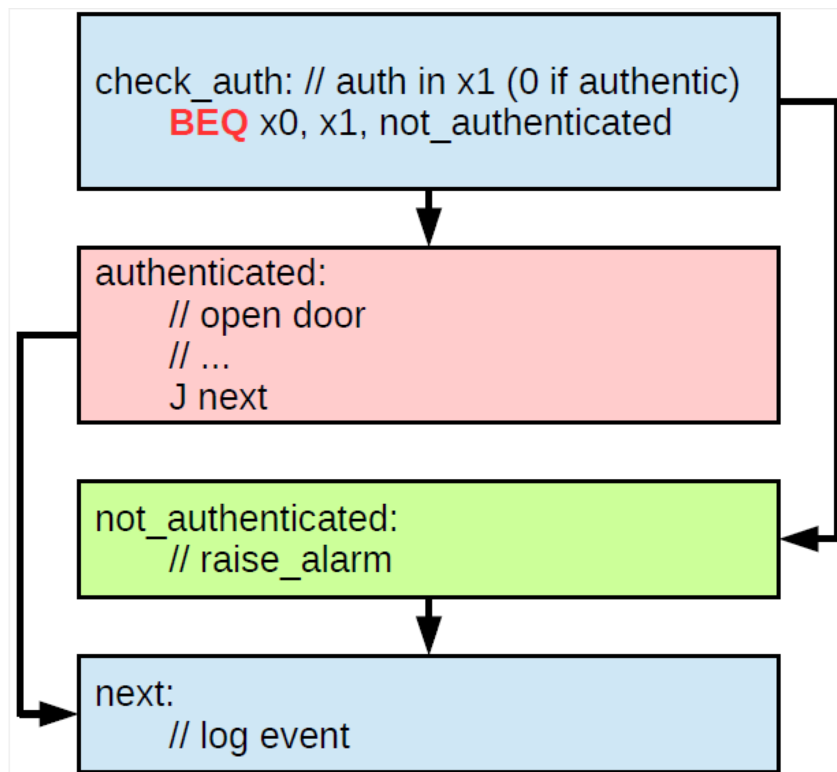
```
unsigned pin = read_pin();
bool auth = check_pin(pin);
if( auth ) {
    open_door();
} else {
    raise_alarm();
}
log_event();
```

Control Flow Integrity is Critical

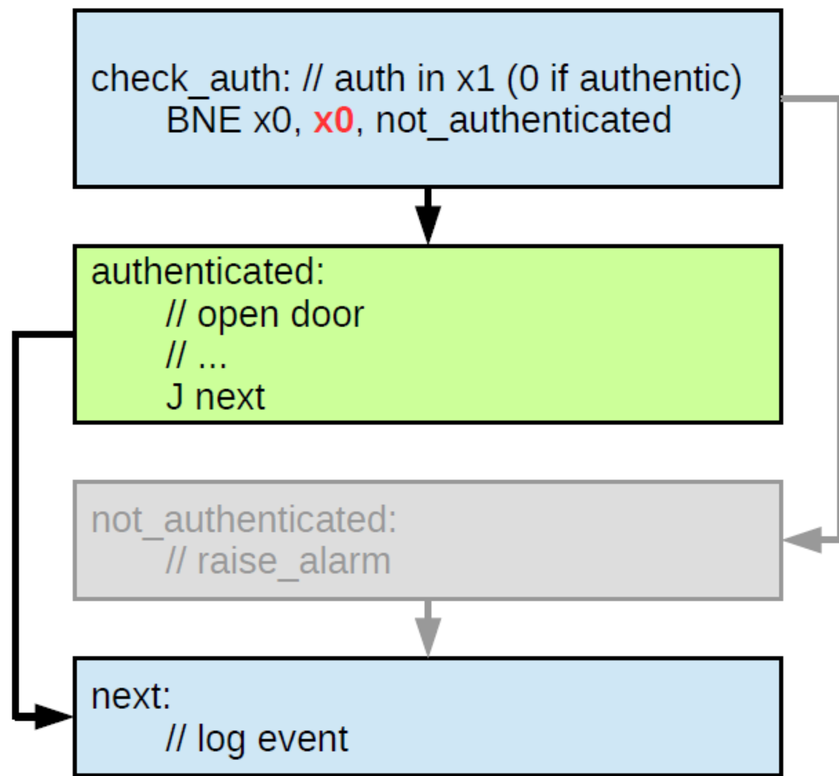


Robert Schilling
Graz University of Technology

Control Flow Integrity is Critical

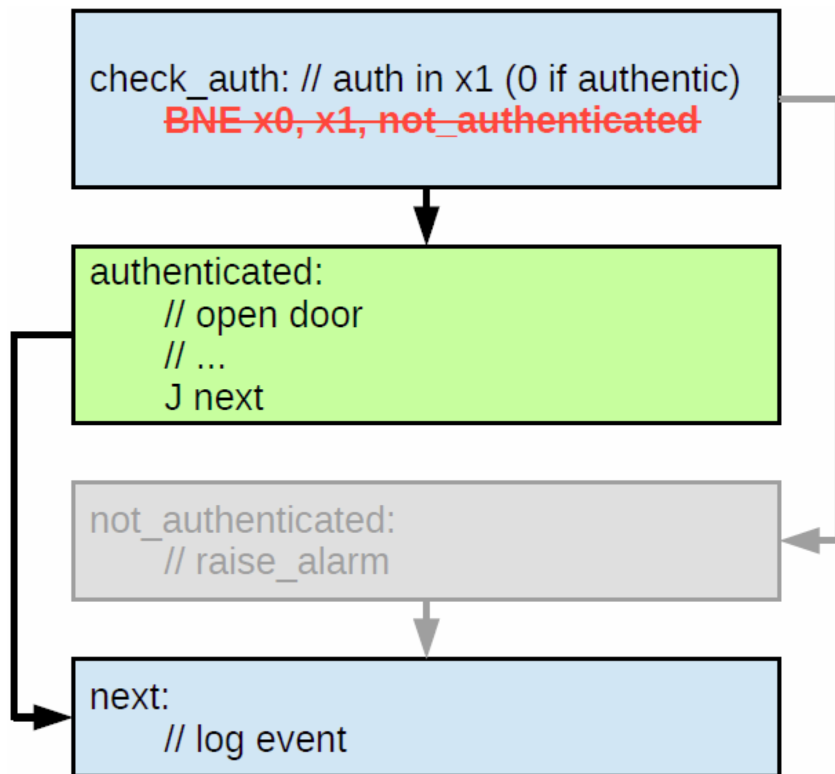


Control Flow Integrity is Critical



Robert Schilling
Graz University of Technology

Control Flow Integrity is Critical



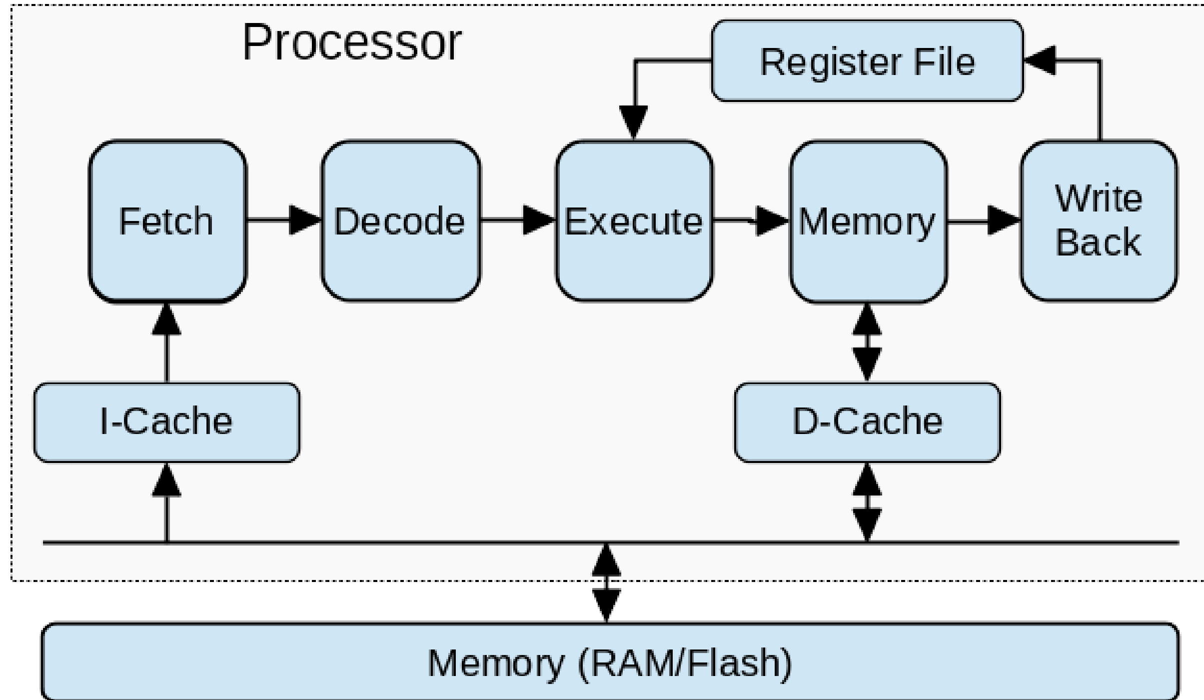
Main Idea

- Sponge-based Control-Flow Protection (SCFP)
  - Hardware supported CFI scheme
  - Encrypts the instruction stream with small granularity
  - Program can only be decrypted correctly as long it is executed correctly

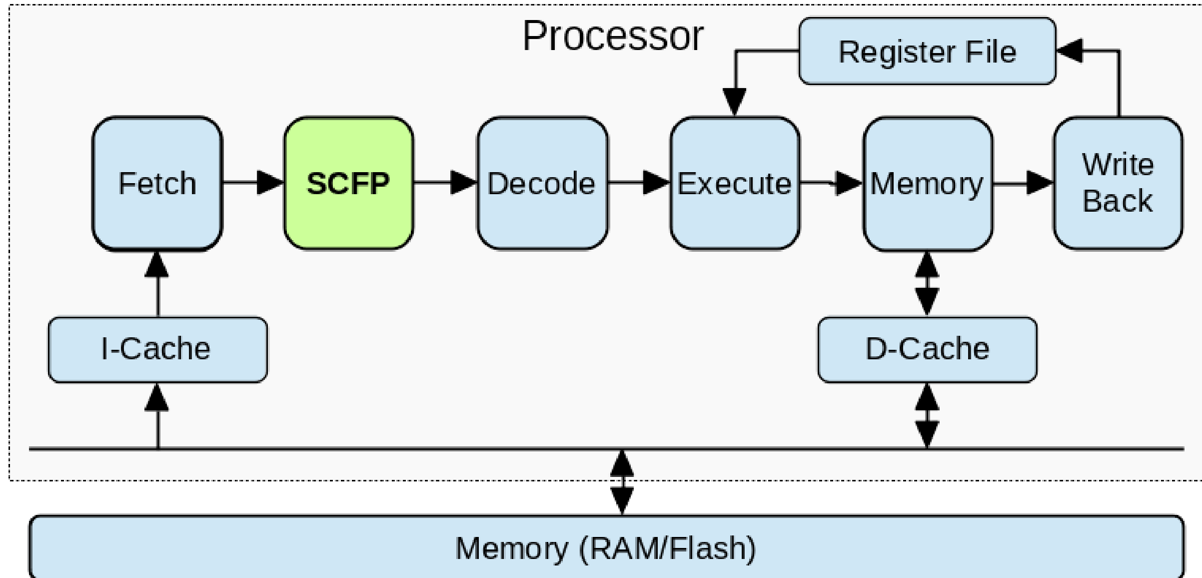
- Costs
  - Highly configurable in terms of security and cost
  - RV32IM AEE-Light: ~10% runtime, ~20% code size

Robert Schilling
Graz University of Technology

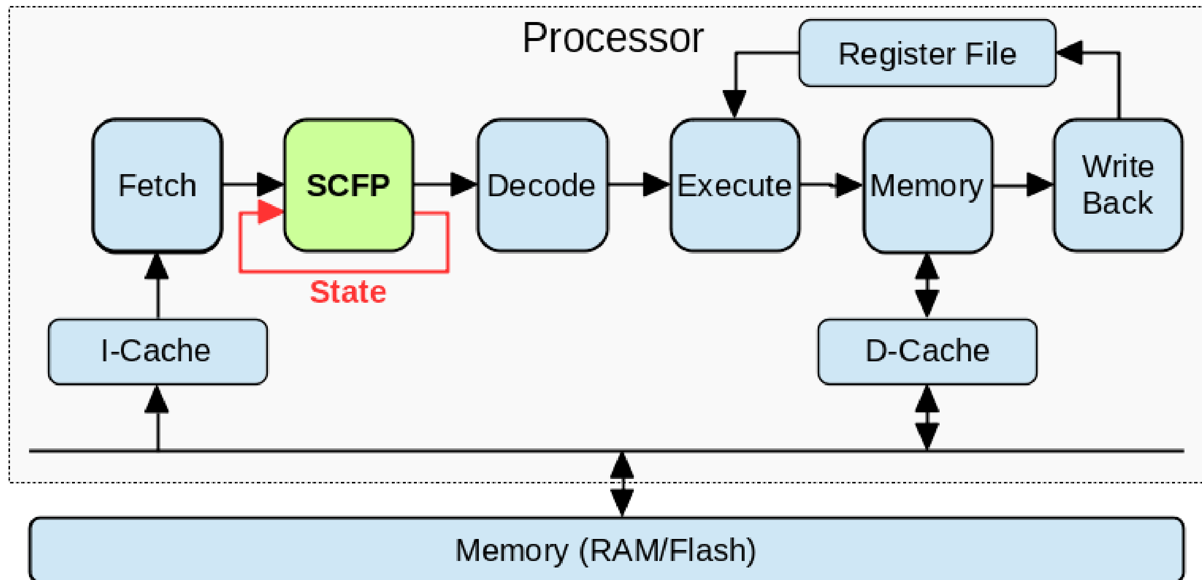
High Level Concept



High Level Concept



High Level Concept



Decryption/Execution Example

```
strcmp

              : ec d0 ee 97
              : 28 ce 77 80
              : 75 41 64 b1
```

```
              : 4b f4 51 75
              : d9 a6 02 ad
              : 51 7d 34 43
```

```
              : 4d 1b c0 0f
              : a3 0f 21 3e
```

Decryption/Execution Example

```
strcmp
0x1b2a0645
                  : ec d0 ee 97
                  : 28 ce 77 80
                  : 75 41 64 b1
```

```
                  : 4b f4 51 75
                  : d9 a6 02 ad
                  : 51 7d 34 43
```

```
                  : 4d 1b c0 0f
                  : a3 0f 21 3e
```

Decryption/Execution Example

```
strcmp
0x1b2a0645
0xdd3fbcce  : 03 06 05 00 : lb a2, 0(a0)
            : 28 ce 77 80
            : 75 41 64 b1
```

```
            : 4b f4 51 75
            : d9 a6 02 ad
            : 51 7d 34 43
```

```
            : 4d 1b c0 0f
            : a3 0f 21 3e
```

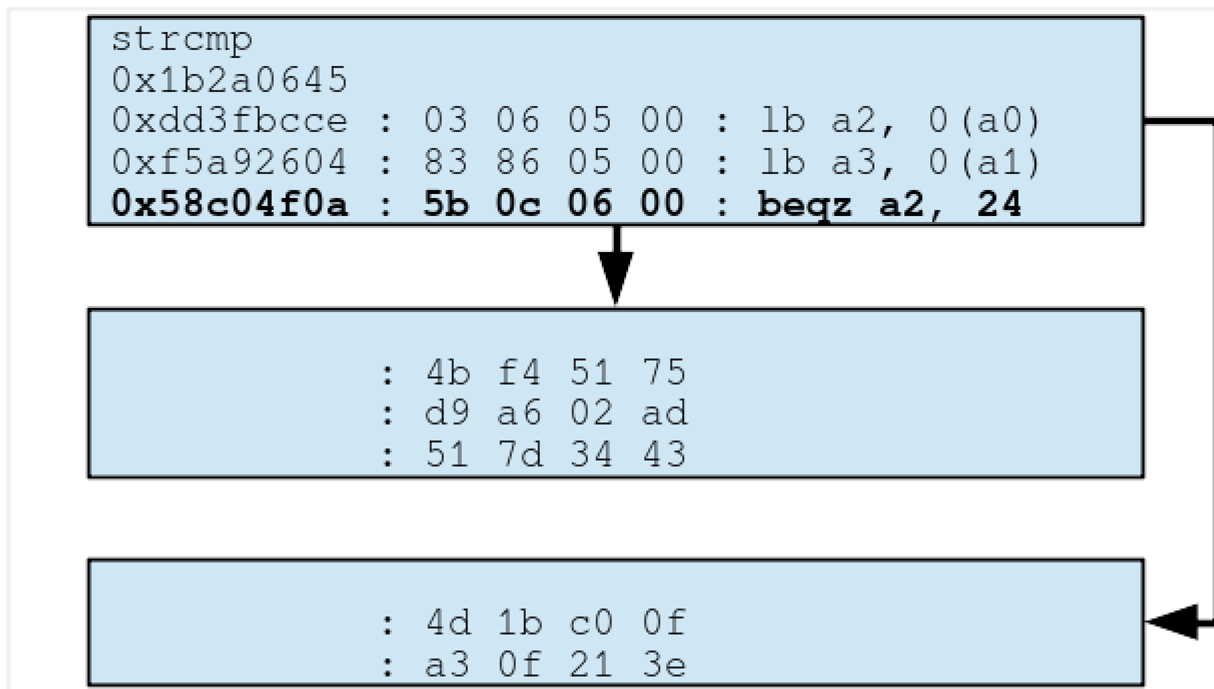
Decryption/Execution Example

```
strcmp
0x1b2a0645
0xdd3fbcce : 03 06 05 00 : lb a2, 0(a0)
0xf5a92604 : 83 86 05 00 : lb a3, 0(a1)
           : 75 41 64 b1
```

```
           : 4b f4 51 75
           : d9 a6 02 ad
           : 51 7d 34 43
```

```
           : 4d 1b c0 0f
           : a3 0f 21 3e
```

Decryption/Execution Example



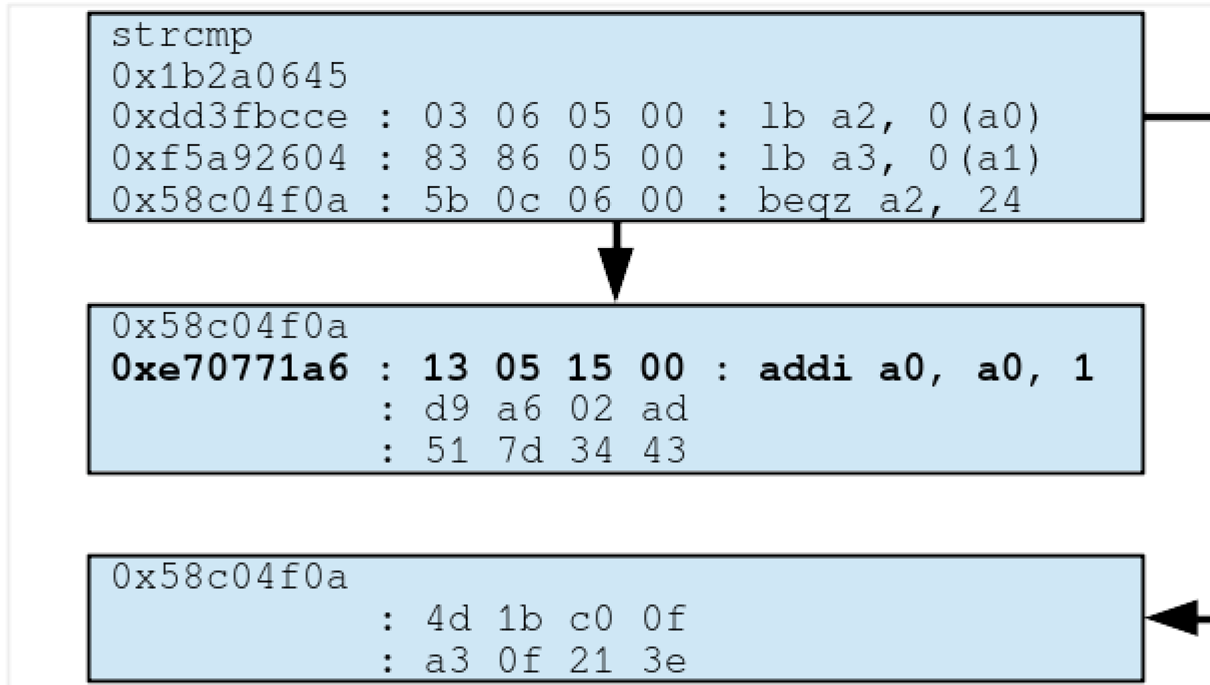
Decryption/Execution Example

```
strcmp
0x1b2a0645
0xdd3fbcce : 03 06 05 00 : lb a2, 0(a0)
0xf5a92604 : 83 86 05 00 : lb a3, 0(a1)
0x58c04f0a : 5b 0c 06 00 : beqz a2, 24
```

```
0x58c04f0a
           : 4b f4 51 75
           : d9 a6 02 ad
           : 51 7d 34 43
```

```
0x58c04f0a
           : 4d 1b c0 0f
           : a3 0f 21 3e
```

Decryption/Execution Example



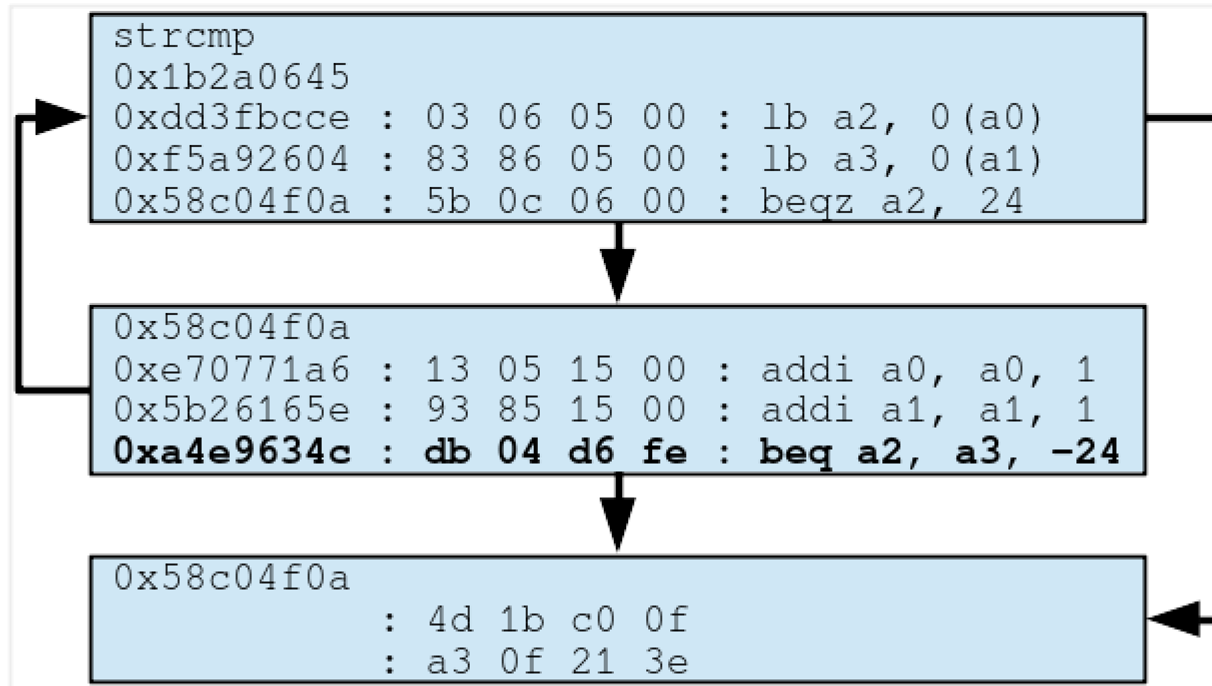
Decryption/Execution Example

```
strcmp
0x1b2a0645
0xdd3fbcce : 03 06 05 00 : lb a2, 0(a0)
0xf5a92604 : 83 86 05 00 : lb a3, 0(a1)
0x58c04f0a : 5b 0c 06 00 : beqz a2, 24
```

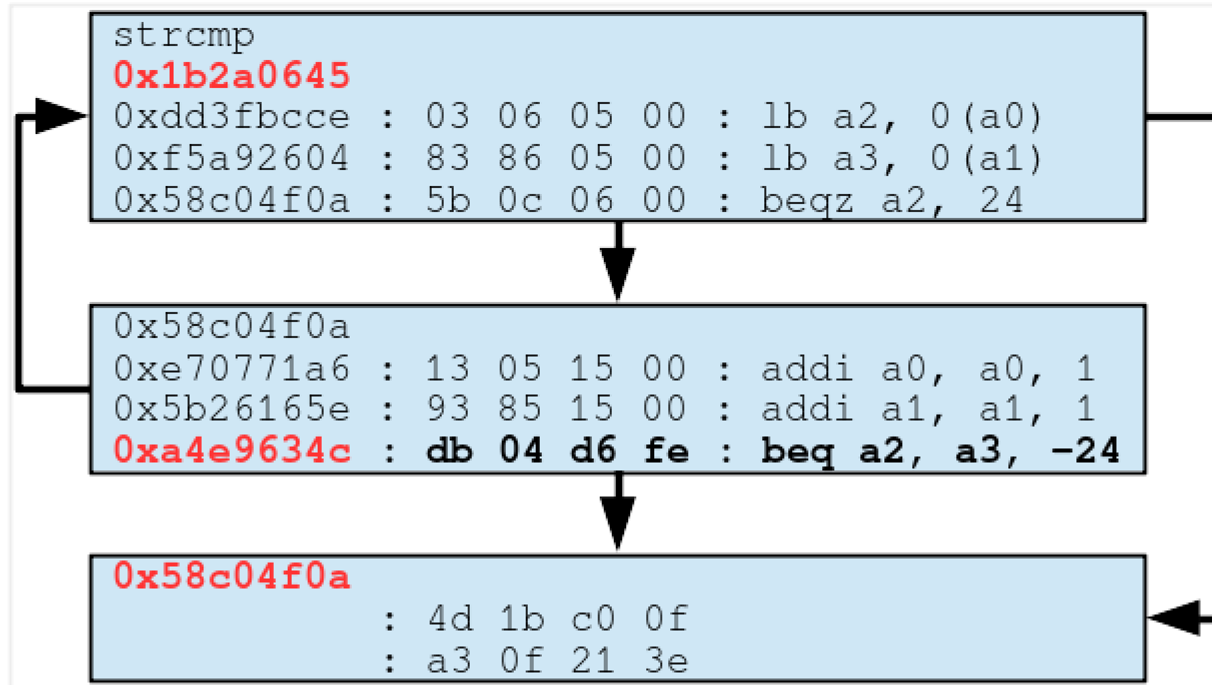
```
0x58c04f0a
0xe70771a6 : 13 05 15 00 : addi a0, a0, 1
0x5b26165e : 93 85 15 00 : addi a1, a1, 1
           : 51 7d 34 43
```

```
0x58c04f0a
           : 4d 1b c0 0f
           : a3 0f 21 3e
```

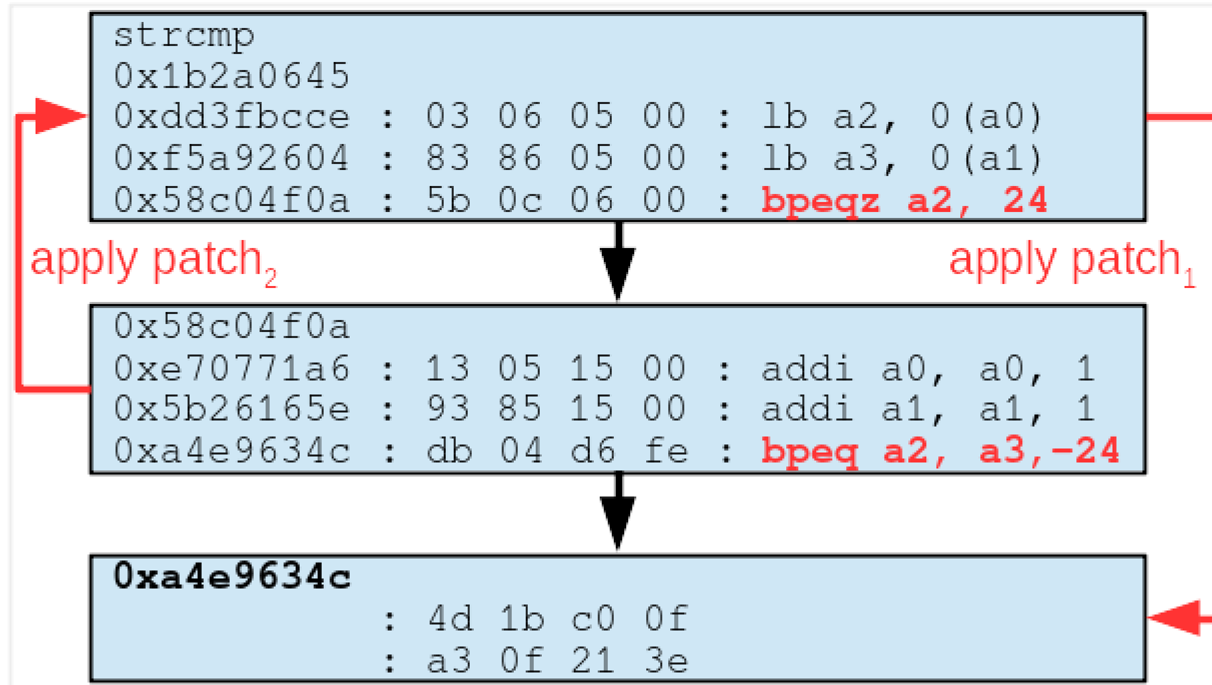
Decryption/Execution Example



Decryption/Execution Example

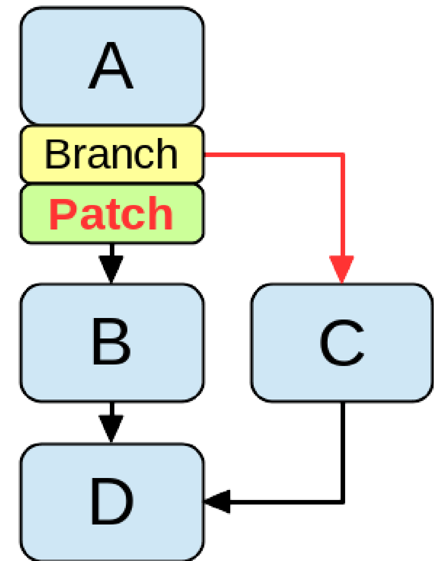


Decryption/Execution Example



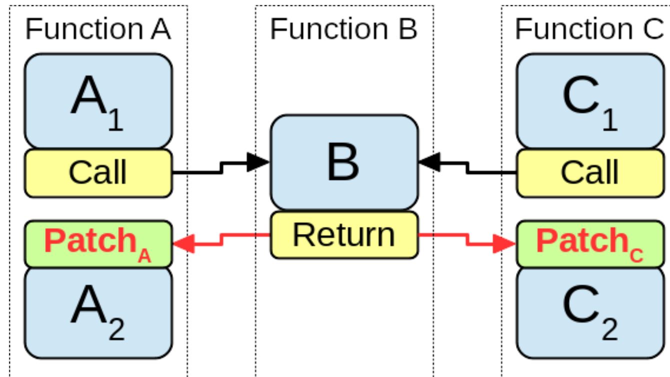
RISC-V ISA Integration - Branches

- Branches additional have an associated patch that is applied conditionally
- New BPEQ, BPNE, BPLT, BPLTU, BPGE, and BPGEU instructions

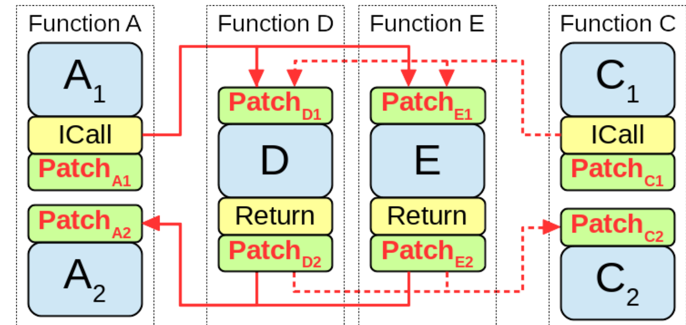


RISC-V ISA Integration – Calls

Direct Calls

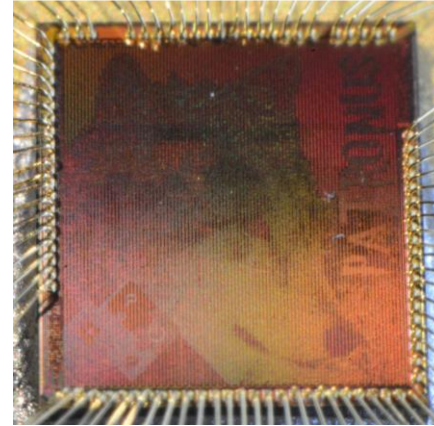


Indirect Calls



Prototype Implementation

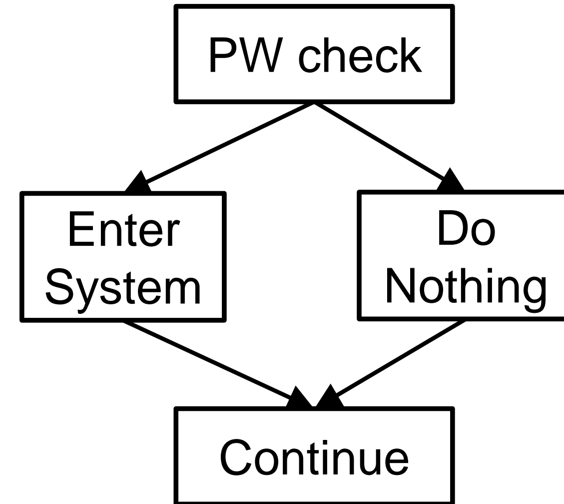
- LLVM-based toolchain
- RI5CY-based hardware
  - AEE-Light with PRINCE in APE-like mode
  - ~30kGE of area for SCFP at 100MHz in UMC65
  - ~10% runtime and ~20% code size overhead



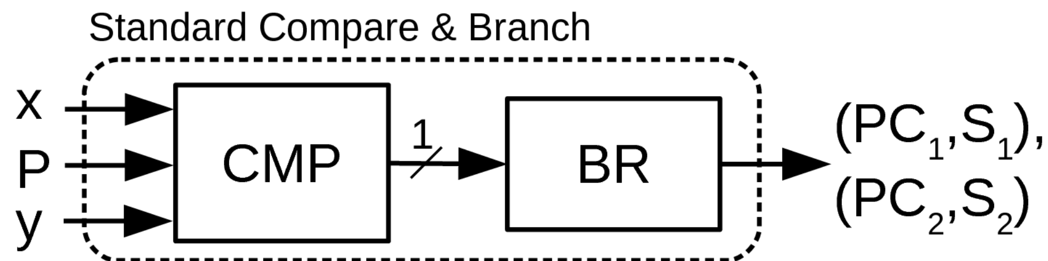
Protected Conditional Branches

Motivation

- Control-flow integrity (CFI) measures restrict the control-flow to valid execution traces
- Branching decisions require extra protection
- Examples of critical applications
  - Password checks, signature verification, …



Classical Conditional Branch

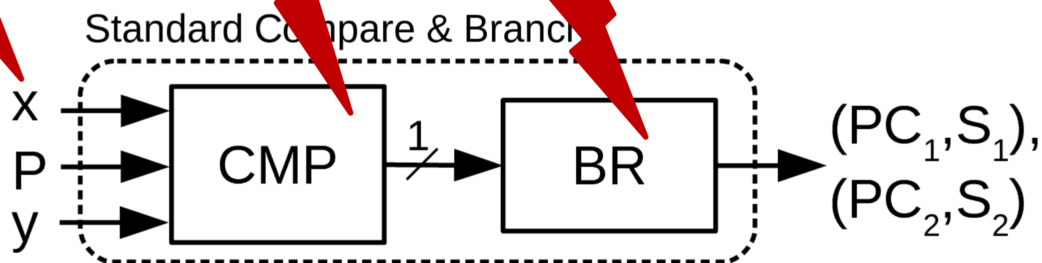


Faulting Conditional Branches

Fault the comparison

Fault the operands

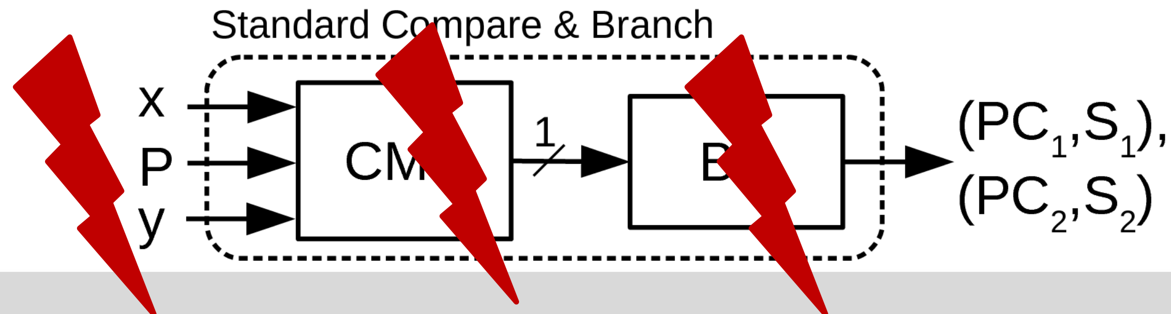
Faulting the branch



Design Goals

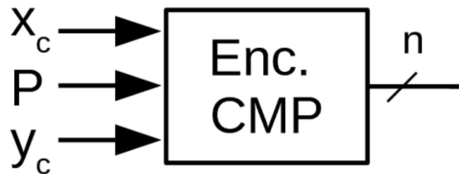
- Maximum flexibility concerning redundancy encoding of data (x, y)
  - Arithmetic codes are efficient for number encoding
  - Linear codes are used for strings

- Minimal changes to hardware



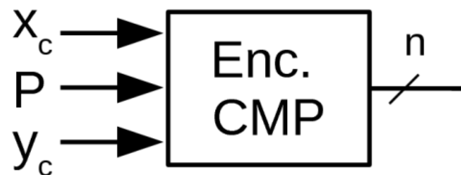
Step 1: Encoded Comparison in Software

- Efficiently possible e.g. for AN Codes
- Output of comparison: n-bit symbol for true or n-bit symbol for false



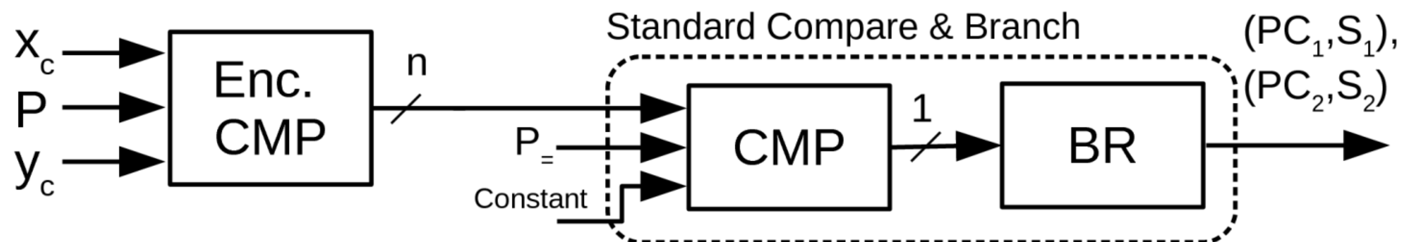
Step 1: Encoded Comparison in Software

How to securely branch based on the n-bit symbol?



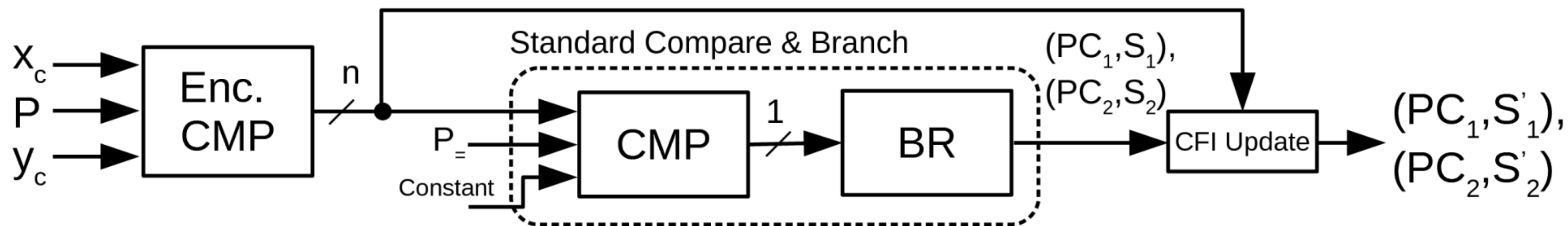
Step 2: Use a standard branch for the actual branching

Use a standard branch for the actual branching ….



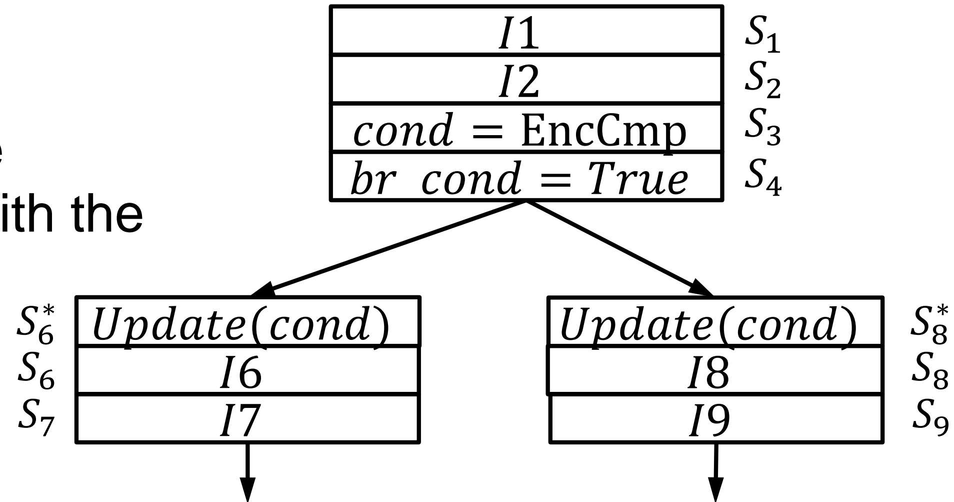
Step 3: Link Comparison Result to CFI State

… and link it to the CFI state.



Example: Protected Conditional Branch

1. Compute the encoded comparison
2. Perform a conditional branch
3. At the branch target: **Link** the redundant **condition** value with the CFI state



Wrong branch and wrong condition lead to invalid CFI state

Prototype Evaluation

- Added new branch instruction to inject first operand to the CFI state

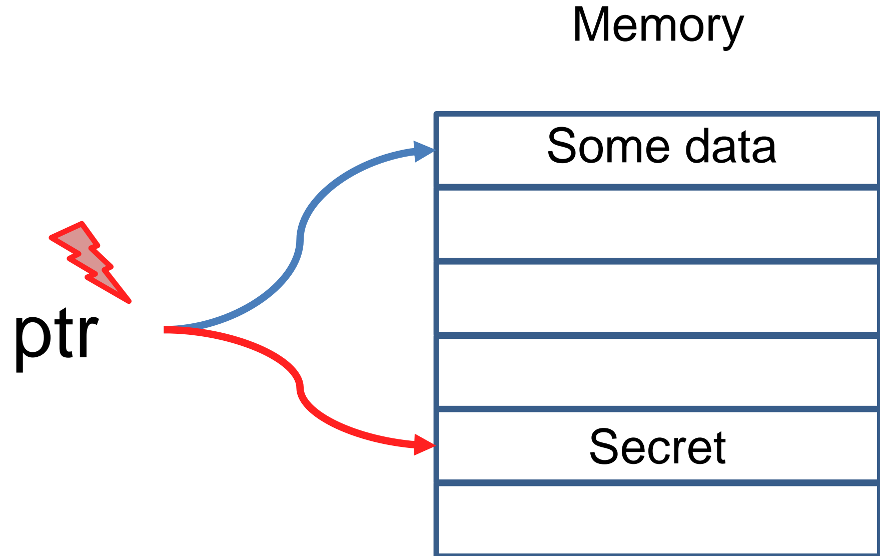
- LLVM-based toolchain
  - Automatically identifies conditional branches
  - Encodes dependent data-flow graph to AN-code domain
  - Inserts software-based comparison algorithm

- Overhead on par with state-of-the-art duplication approaches

Secure Memory Access

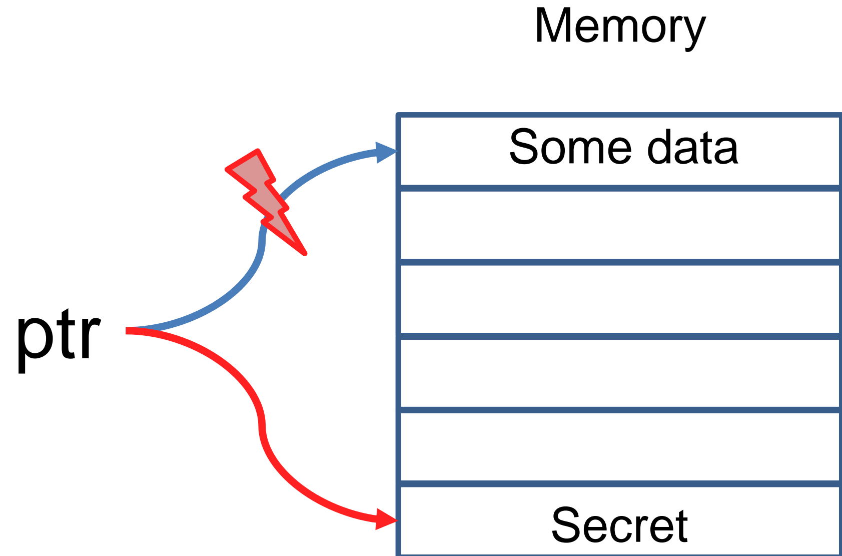
Faulting the Pointer

- Faulted pointer redirects the memory access



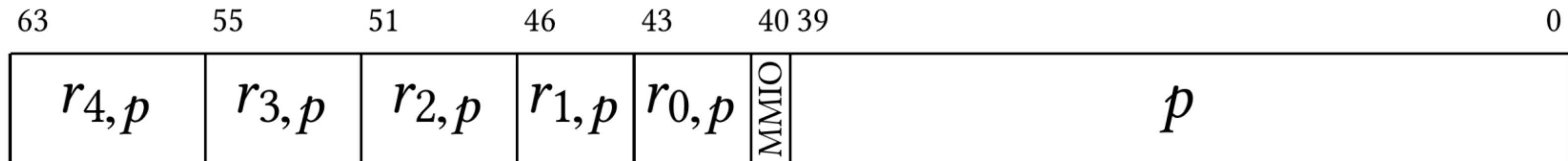
Faulting the Memory Access

- Faulted pointer redirects the memory access
- Faulting the memory access itself leads to a wrong access



Pointer Protection with Residue Codes

- Use multi-residue code to protect the pointer
  - Gives direct access to the functional value → no expensive decoding required
  - Supports pointer arithmetic
- Redundancy stored in the pointer

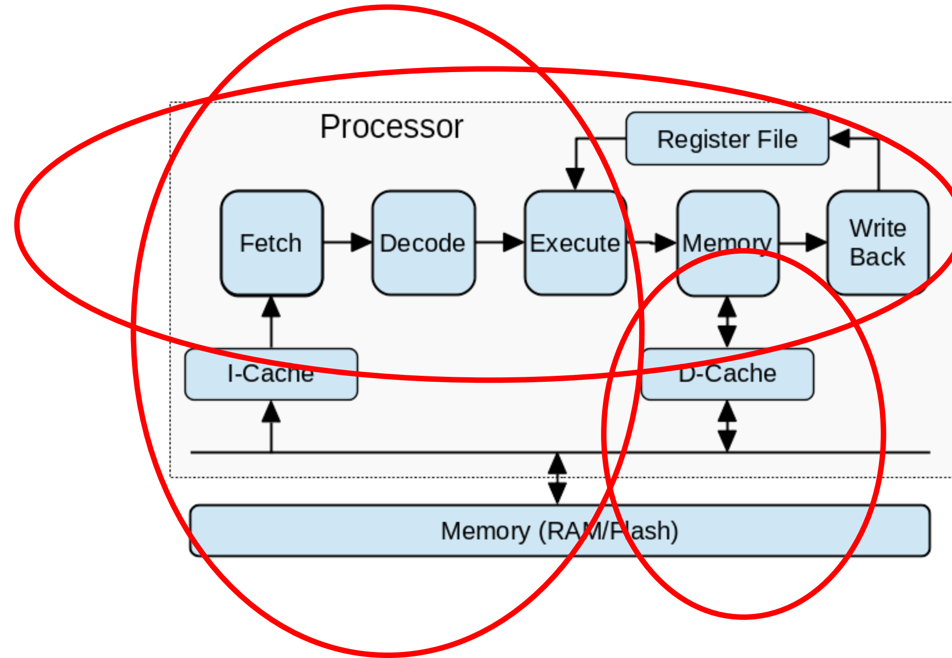


Protecting the Memory Accesses

- Write memory in the form $mem[p] = l_p(D_{Reg})$
- Inverse to read data back $D_{Reg} = l_p^{-1}(mem[p])$
- Xor operation → chosen for low-overhead
  - $mem[p] = p \oplus D_{Reg}, \quad D_{Reg} = p \oplus mem[p]$
  - **Problems** with granularity
- Use a byte-wise linking granularity to support arbitrary accesses

Prototype Evaluation

- FPGA prototype based PULP by ETH Zurich with 5% area overhead
- ISA extension residue arithmetic and linked memory accesses
- Custom LLVM compiler prototype transforms all pointers
- Transformed all data pointers, protected all pointer arithmetic, replaced all memory accesses
- ~7% runtime and ~10% code size overhead


Processor
Register File
Fetch
Decode
Execute
Memory
Write
Back
I-Cache
D-Cache
Memory (RAM/Flash)

We are hiring PhD students, postdocs, senior researchers, faculty

Stefan.Mangard@iaik.tugraz.at

https://Cybersecurity-Campus.tugraz.at