MINES Saint-Étienne

Une école de l'IMT

ARSENE

Resilience of RISC-V processors against fault injections

Théophile Gousselot
Ph.D. Student
Arsene - Campus Cyber

Feb 7, 2024

INSPIRING INNOVATION SINCE 1816

Work in progress: Harden RISC-V core

Work in progress: Harden RISC-V core

PEPR Arsene:

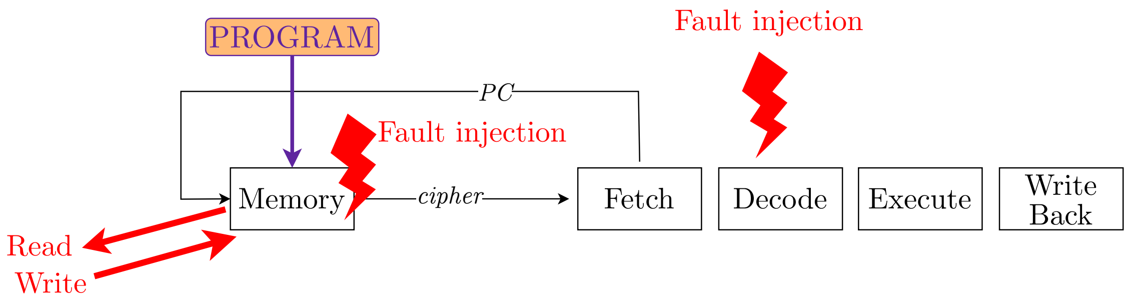
`Tâche 1.1 : RISC-V 32-bits sécurisé contre les FI`

Security properties

- ➜ **Integrity** of Control Flow and Control Signals
- ➜ **Confidentiality** of Instructions

Work in progress: Harden RISC-V core

Threat model



Work in progress: Harden RISC-V core
Bibliography

Existing state-of-the-art countermeasures:

- SOFIA / SPONGE / MAFIA / CONFIDAENT / HAPEI

Fault injection on RISC-V architecture:

- Johan Laurent et al. / Simon Tollec et al. / Anthony Zgheib et al.

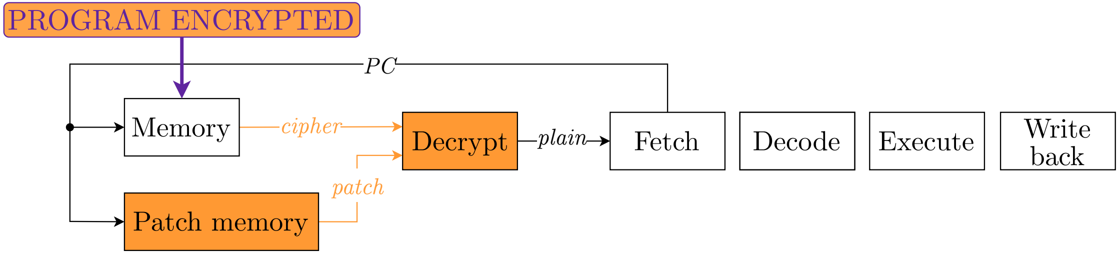
Work in progress: Harden RISC-V core
Solution

**Chained and Authenticated Encryption of Instructions,
with Control Signal association.**

Work in progress: Harden RISC-V core

Solution

Chained and Authenticated Encryption of Instructions, with Control Signal association.

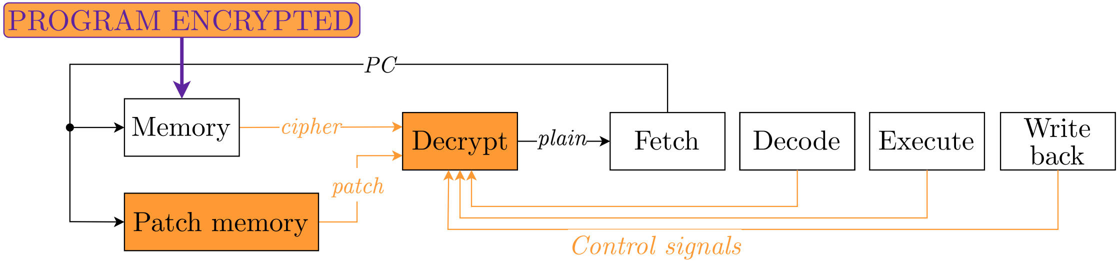


1. Chained Encryption of Instructions (before programming memory)
2. On-the-fly Decryption

Work in progress: Harden RISC-V core

Solution

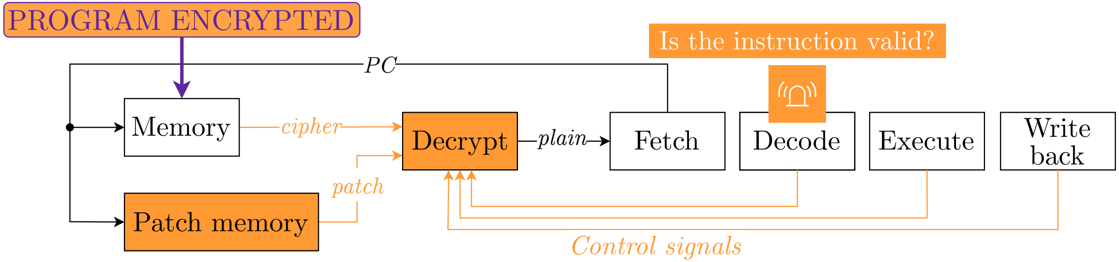
Chained and Authenticated Encryption of Instructions, with Control Signal association.



1. Chained Encryption of Instructions (before programming memory)
2. On-the-fly Decryption

Work in progress: Harden RISC-V core
Solution

Chained and Authenticated Encryption of Instructions, with Control Signal association.

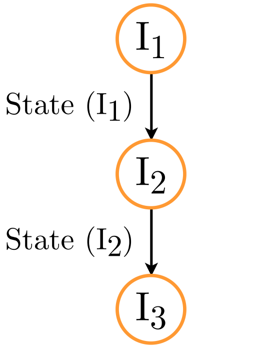


1. Chained Encryption of Instructions (before programming memory)
2. On-the-fly Decryption

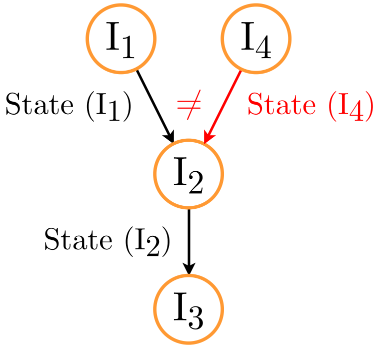
Work in progress: Harden RISC-V core

Solution

Need for patches



Linear decryption

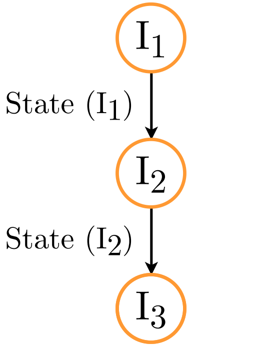


Decryption when branches merge

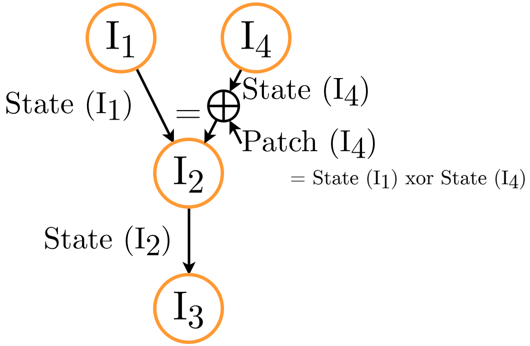
Work in progress: Harden RISC-V core

Solution

Need for patches



Linear decryption



Decryption when branches merge

Countermeasure Description

Countermeasure Description
RISC-V core: cv32e40p

OpenHW group:

- **cv32e40p** 4 stages core - rv32imc
- **core-v-verif** RTL simulation
- **core-v-mcu** microcontroller - FPGA

Countermeasure Description
Authenticated Encryption: ASCON

ASCON: cipher suite, which provides Authenticated Encryption with Associated Data

Countermeasure Description
Authenticated Encryption: ASCON

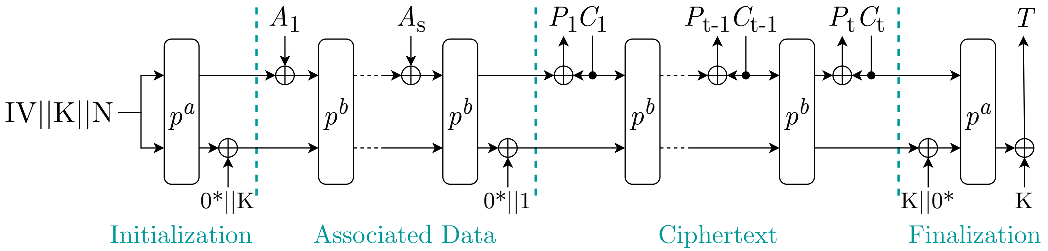
ASCON: cipher suite, which provides Authenticated Encryption with Associated Data

- Winner of CAESAR Authenticated Encryption - Lightweight Cryptography (2019)
- Winner of NIST - Lightweight Cryptography (2023) $\Rightarrow$ **standardize the ASCON family**
- Lightweight (better Throughput per Area than AES [1])
- Highly tested
- Large security margins

[1] *"Need for Low-latency Ciphers: A Comparative Study of NIST LWC Finalists"*, NIST LWC Workshop 2022, Tolga Yalcin - Google

Countermeasure Description
Authenticated Encryption: ASCON

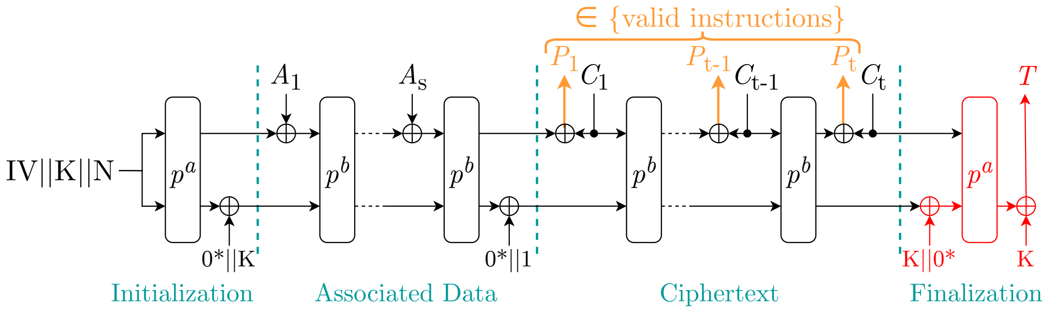
ASCON: Authenticated Encryption provided from Tag (T)



ASCON Decryption scheme

Countermeasure Description
Authenticated Encryption: ASCON

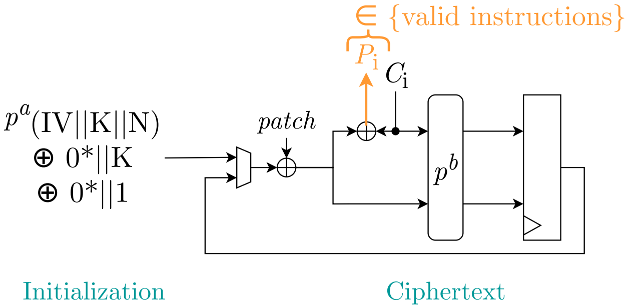
ASCON: Use-case of restricted set of valid data



ASCON Decryption scheme

Countermeasure Description
Authenticated Encryption: ASCON

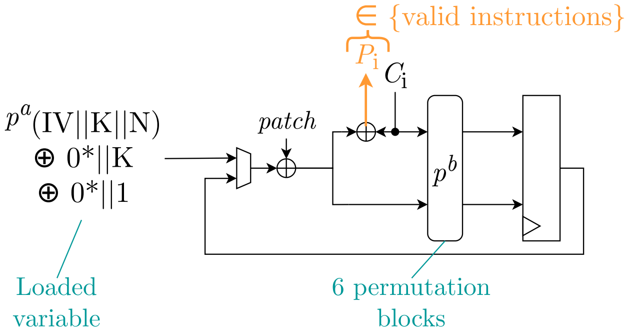
ASCON: Authenticated Encryption provided from restricted set of valid instructions



ASCON Decryption scheme

Countermeasure Description
Authenticated Encryption: ASCON

ASCON: Authenticated Encryption provided from restricted set of valid instructions


$\in$ {valid instructions}
P_i
C_i
$p^a(\mathrm{IV}||\mathrm{K}||\mathrm{N})$
$\oplus\ 0^*||\mathrm{K}$
$\oplus\ 0^*||1$
patch
p^b
Loaded variable
6 permutation blocks

ASCON Decryption scheme

Countermeasure Description
Patch policy

- Patches stored in external memory
- Patches addressed by Program Counter
- ... and applied according to Control Signals

➜ No need for custom instructions $\Rightarrow$ zero clock cycle overhead

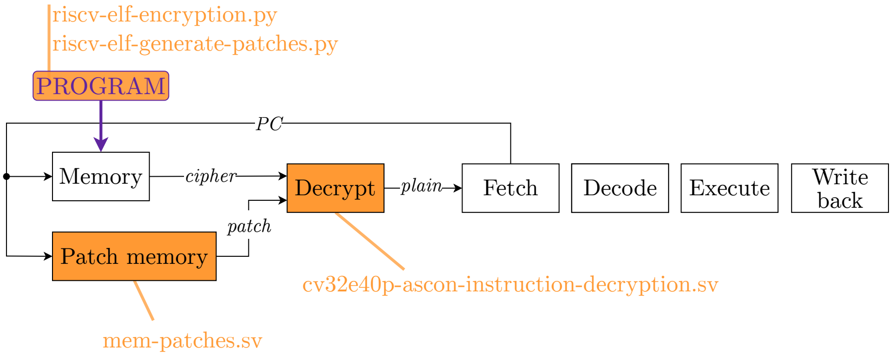
Countermeasure Description
Solution Flow (without Control Signals)

Soft:

1. Encrypt instructions sequentially
2. Build CFG
3. Generate patches

Hard:

1. Patch memory
2. ASCON decryption

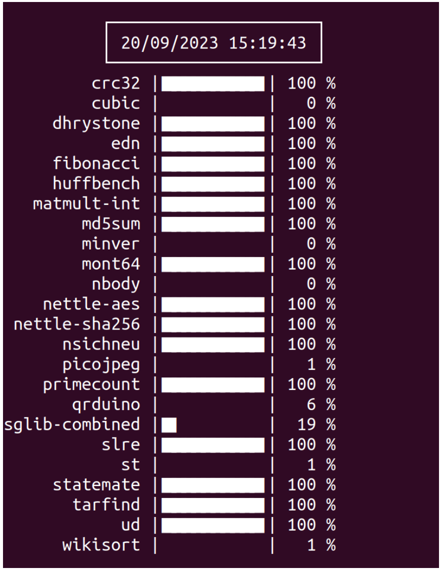


Countermeasure Validation

Countermeasure Validation
Behavioral Simulation

- core-v-verif
- Verilator
- Embench
  - ✔ valid execution for 15/23 programs

Indirect jump destinations are considered known



```
          20/09/2023 15:19:43

        crc32 |██████████| 100 %
        cubic |          |   0 %
    dhrystone |██████████| 100 %
          edn |██████████| 100 %
    fibonacci |██████████| 100 %
    huffbench |██████████| 100 %
  matmult-int |██████████| 100 %
       md5sum |██████████| 100 %
       minver |          |   0 %
       mont64 |██████████| 100 %
        nbody |          |   0 %
   nettle-aes |██████████| 100 %
nettle-sha256 |██████████| 100 %
     nsichneu |██████████| 100 %
     picojpeg |          |   1 %
   primecount |██████████| 100 %
      qrduino |          |   6 %
sglib-combined |██        |  19 %
         slre |██████████| 100 %
           st |          |   1 %
    statemate |██████████| 100 %
     tarfind |██████████| 100 %
           ud |██████████| 100 %
     wikisort |          |   1 %
```

Countermeasure Validation
FPGA

- ➜ **Goals:** Validation & Looking for maximal frequency and utilization
- ✖ core-v-mcu (F=10MHz) $\rightarrow$ Homemade core-v-fpga
- ● Vivado flow: Nexys Video (Artix 7)

Countermeasure Validation
FPGA

➜ **Goals:** Validation & Looking for maximal frequency and utilization

✖ core-v-mcu (F=10MHz) → Homemade core-v-fpga

● Vivado flow: Nexys Video (Artix 7)

	core-v-fpga		**core-v-fpga enc**
Freq$_{max}$	67.45 MHz	$\Rightarrow$	47.25 MHz
Period$_{min}$	14.8 ns	$\Rightarrow$	21.2 ns
Cycles	N	$\Rightarrow$	N
LUT	4065	$\Rightarrow$	7534
FF	2065	$\Rightarrow$	3377
BRAM	32	$\Rightarrow$	68

MINES Saint-Étienne
Une école de l'IMT

Countermeasure Validation
FPGA

- ➜ **Goals:** Validation & Looking for maximal frequency and utilization
- ✖ core-v-mcu (F=10MHz) $\rightarrow$ Homemade core-v-fpga
- ● Vivado flow: Nexys Video (Artix 7)

	core-v-fpga		core-v-fpga enc
$\textbf{Freq}_{max}$	67.45 MHz	$\Rightarrow$	47.25 MHz
$\textbf{Period}_{min}$	14.8 ns	$\Rightarrow$	21.2 ns
Cycles	N	$\Rightarrow$	N
LUT	4065	$\Rightarrow$	7534
FF	2065	$\Rightarrow$	3377
BRAM	32	$\Rightarrow$	68

Enhancements:

Split critical paths

Roll permutations:
$\texttt{clk}_{ascon} = \texttt{M} \times \texttt{clk}_{core}$

Countermeasure Validation

FPGA

- **Goals:** Validation & Looking for maximal frequency and utilization
- ✖ core-v-mcu (F=10MHz) → Homemade core-v-fpga
- ● Vivado flow: Nexys Video (Artix 7)

	core-v-fpga		core-v-fpga enc
Freq$_{max}$	67.45 MHz	⇒	47.25 MHz
Period$_{min}$	14.8 ns	⇒	21.2 ns
Cycles	N	⇒	N
LUT	4065	⇒	7534
FF	2065	⇒	3377
BRAM	32	⇒	68

Enhancements:

Split critical paths

Roll permutations:

$\texttt{clk}_{ascon} = \texttt{M} \times \texttt{clk}_{core}$

MINES Saint-Étienne
Une école de l'IMT

Countermeasure Validation
FPGA

- ➜ **Goals:** Validation & Looking for maximal frequency and utilization
- ✖ core-v-mcu (F=10MHz) $\rightarrow$ Homemade core-v-fpga
- ● Vivado flow: Nexys Video (Artix 7)

	core-v-fpga		**core-v-fpga enc**
Freq$_{max}$	67.45 MHz	$\Rightarrow$	47.25 MHz
Period$_{min}$	14.8 ns	$\Rightarrow$	21.2 ns
Cycles	N	$\Rightarrow$	N
LUT	4065	$\Rightarrow$	7534
FF	2065	$\Rightarrow$	3377
BRAM	32	$\Rightarrow$	68

Enhancements:

Split critical paths

Roll permutations:
$\texttt{clk}_{ascon} = \texttt{M} \times \texttt{clk}_{core}$

Reduce patch size or change patch policy

Countermeasure Validation

Security

Experiments: VerifyPin

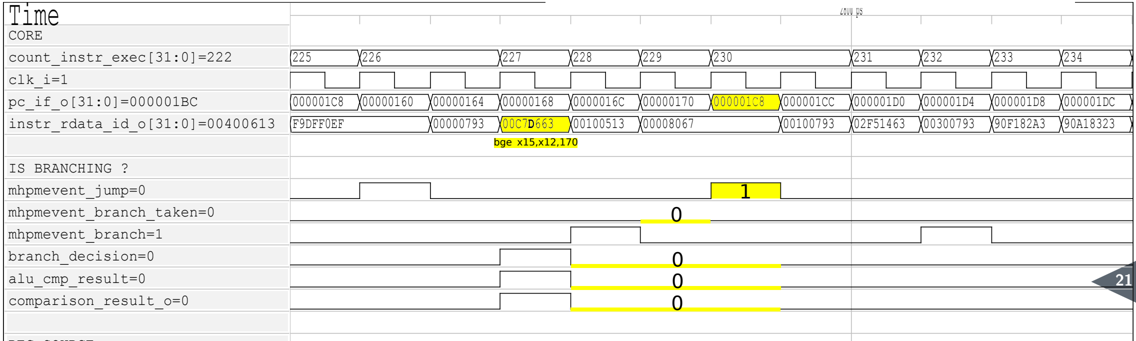
Execution: Plain

Fault: memory (164: blt → bge)

Authenticated: ✔

Detection:

```
  164: 00c7c663      blt x15,x12,170     //comparison end
  168: 00100513      addi x10,x0,1           x10 = a1
✔ 16c: 00008067      jalr x0,0(x1)           x11 = a2
  170: 00f506b3      add x13,x10,x15         x12 = size
  174: 00f58733      add x14,x11,x15         x13 = a1+i
  178: 0006c683      lbu x13,0(x13)          x14 = a2+i
  17c: 00074703      lbu x14,0(x14)          x15 = i
  180: 00e69663      bne x13,x14,18c     //compare 4bits
  184: 00178793      addi x15,x15,1
  188: fddff06f      jal x0,164
  18c: 00000513      addi x10,x0,0
✖ 190: 00008067      jalr x0,0(x1)
```



```
Time                                  2800 ps
CORE
count_instr_exec[31:0]=222     225      226       227       228       229       230       231       232       233       234
clk_i=1
pc_if_o[31:0]=000001BC         000001C8 00000160  00000164  00000168  0000016C  00000170  000001C8  000001CC  000001D0  000001D4  000001D8  000001DC
instr_rdata_id_o[31:0]=00400613 F9DFF0EF          00000793  00C7D663  00100513  00008067            00100793  02F51463  00300793  90F182A3  90A18323
                                                            bge x15,x12,170
IS BRANCHING ?
mhpmevent_jump=0                                                                1
mhpmevent_branch_taken=0                                                0
mhpmevent_branch=1
branch_decision=0                                                       0
alu_cmp_result=0                                                        0
comparison_result_o=0                                                   0
```

Countermeasure Validation

Security

Experiments: VerifyPin

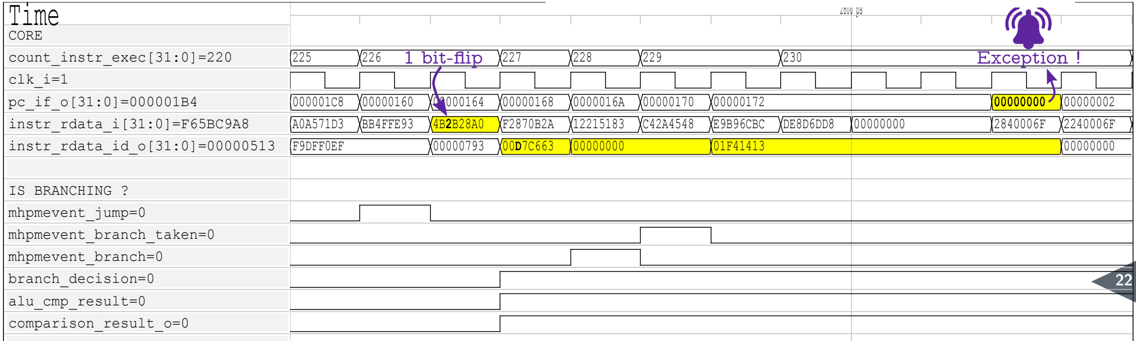
Execution: Encrypted

Fault: memory (164: blt → bge)

Authenticated:

Detection: ✓

```
    164: 00c7c663          blt x15,x12,170     //comparison end
    168: 00100513          addi x10,x0,1           x10 = a1
 ✓  16c: 00008067          jalr x0,0(x1)           x11 = a2
    170: 00f506b3          add x13,x10,x15         x12 = size
    174: 00f58733          add x14,x11,x15         x13 = a1+i
    178: 0006c683          lbu x13,0(x13)          x14 = a2+i
    17c: 00074703          lbu x14,0(x14)          x15 = i
    180: 00e69663          bne x13,x14,18c     //compare 4bits
    184: 00178793          addi x15,x15,1
    188: fddff06f          jal x0,164
    18c: 00000513          addi x10,x0,0
 ✗  190: 00008067          jalr x0,0(x1)
```



Conclusion

Conclusion

Done:

- ✔ **Soft:** Instruction encryption and patch generation
- ✔ **Hard:** Instruction decryption (47.25 MHz)
- ✔ **Validation:** Behavioral, FPGA
- ✔ **Attack simulation:** a few scenarios to illustrate the concept

Conclusion

Done:

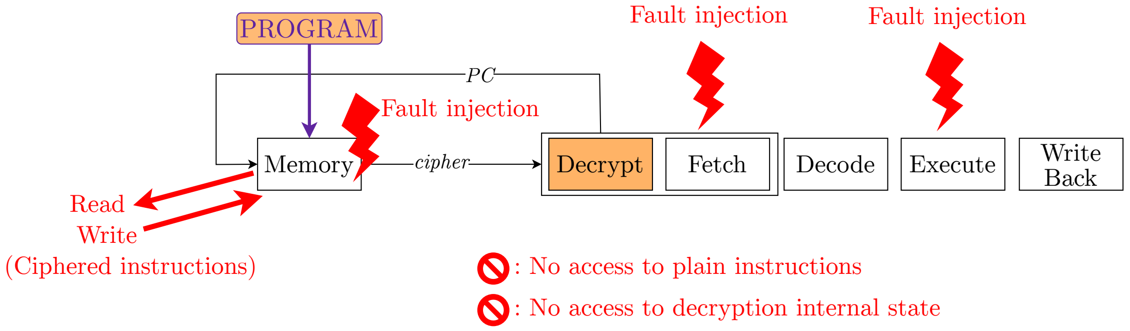
- ✓ **Soft:** Instruction encryption and patch generation
- ✓ **Hard:** Instruction decryption (47.25 MHz)
- ✓ **Validation:** Behavioral, FPGA
- ✓ **Attack simulation:** a few scenarios to illustrate the concept

In progress:

- ➜ **Rolled permutations:** LUT utilization ↘
- ● **Control Signals association (only static):** Signal integrity
- ● **Reduce patch size:** BRAM utilization ↘
- ● **Other patch policy:** BRAM utilization ↘

Extra slides

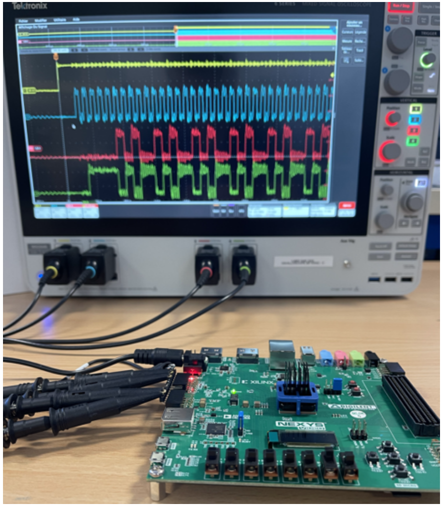
Extra slides



Linear Code Extraction: Hardware Demonstration

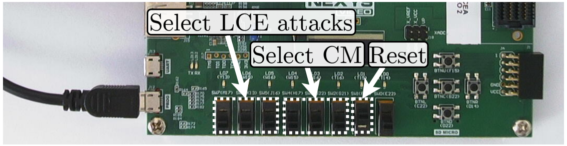
Linear Code Extraction: Hardware Demonstration

Théophile Gousselot | SAS | HW DEMO HOST 2023



FPGA-Based demonstration

- **Purpose:** Interactive demonstration on a booth
- Real-time LCE (Emulated microprobing)
- Quickly assess and observe core behavior
- Nexys Video, Artix 7
- core-v-mcu microcontroller



RISC-V Environment

RISC-V Environment

Théophile Gousselot | SAS

RISC-V core: cv32e40p (OpenHW Group)

RISC-V Compilation: gcc + binary parsing **[Python scripts]**

- ✓ Parse, edit binary and build CFG
- ✓ ASCON encryption

RISC-V verification: core-v-verif (OpenHW Group) + improvements for security validation **[Makefile and scripts]**

- ✓ Behavioral simulation with verilator (Embench programs)
- ✓ Analyze of program execution traces (PC, instr in fetch, etc.)
- ✓ Force signals values for specific cycles
- ✗ UVM Verification methodology not setup but available

RISC-V Environment
Théophile Gousselot | SAS

FPGA flow: core-v-mcu (OpenHW Group)

- ✔ Microcontroller used to run the HOST LCE demo

FPGA flow: core-v-fpga (Made by ourselves) **[Makefile, TCL scripts]**

- ✔ Homemade small microcontroller (cv32e40p, instr/data memory, interface (=wrapper))
  - ➜ 10 times smaller than core-v-mcu
  - ➜ core frequency 6 times higher than core-v-mcu
- ✔ **non-project-mode flow** for behavioral simulation
  - ➜ Can run every Embench program (generate VCD, generate Trace of PC/instr)
- ✔ **project-mode flow** for synthesis, implementation, bitstream generation
  - ➜ Automated project creation by TCL scripts (including Clock Wizard, Integrated Logic Analyzer for debug purpose)
  - ➜ TCL script to find a design maximal frequency

MINES Saint-Étienne

Une école de l'IMT

Resilience of RISC-V processors against fault injections

Théophile Gousselot
Ph.D. Student
Arsene - Campus Cyber
Feb 7, 2024

INSPIRING INNOVATION SINCE 1816