



MINES
Saint-Étienne

Une école de l'IMT



ARSENE

ÉCOLE NATIONALE SUPÉRIEURE DES MINES

FOURNEYRON
1891 - 1927

Code Encryption for Confidentiality and Execution Integrity down to Control Signals

Théophile Gousselot

Ph.D. Student

SemSecuElec - Rennes, Inria

Oct 18, 2024



Who am I?

- 1 Who am I?
- 2 Context for execution integrity
- 3 Proposed scheme
- 4 Validation and characterization
- 5 Conclusion

Who am I?



github.com/theophile-gousselot

Master's Degree in Microelectronics and Computer Science

- Embedded system security
- Final year internship at STMicroelectronics: Control Flow Integrity



Master's Degree in Microelectronics and Computer Science

- Embedded system security
- Final year internship at STMicroelectronics: Control Flow Integrity



github.com/theophile-gousselot



PhD Student: Harden RISC-V cores

- **Secure Architectures and Systems** department (Gardanne)
- Linear Code Extraction
- Execution Integrity



Master's Degree in Microelectronics and Computer Science

- Embedded system security
- Final year internship at STMicroelectronics: Control Flow Integrity



github.com/theophile-gousselot



PhD Student: Harden RISC-V cores

→ **Secure Architectures and Systems** department (Gardanne)

● Linear Code Extraction

● Execution Integrity

Today!



Context for execution integrity

1 Who am I?

2 Context for execution integrity

- Microcontroller as our main embedded system to protect
- Abstraction hierarchy
- Threat model
- Goal
- Control Flow and Instruction Integrity
- Control Signal Integrity
- State-of-the-art execution integrity

3 Proposed scheme

4 Validation and characterization

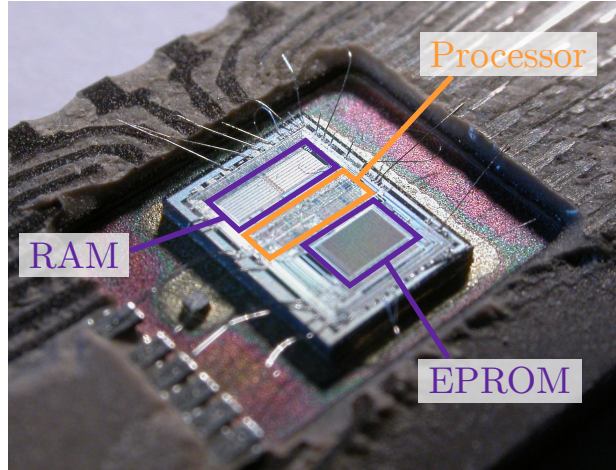
5 Conclusion

Context for execution integrity

Microcontroller as our main embedded system to protect

Microntroller: one die...

- Processor
- RAM
- ROM



Context for execution integrity

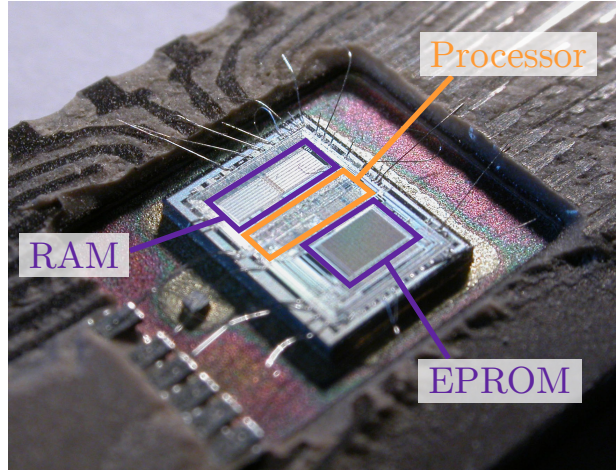
Microcontroller as our main embedded system to protect

Microntroller: one die...

- Processor
- RAM
- ROM

Program:

- ✘ Operating System¹
- ✔ Bare-metal (specialized task)



¹no memory management unit

Context for execution integrity

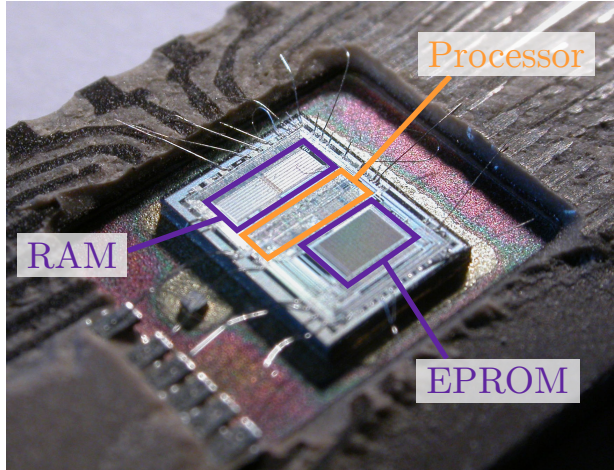
Microcontroller as our main embedded system to protect

Microntroller: one die...

- Processor
- RAM
- ROM

Program:

- ✘ Operating System¹
- ✔ Bare-metal (specialized task)



¹no memory management unit

Context for execution integrity

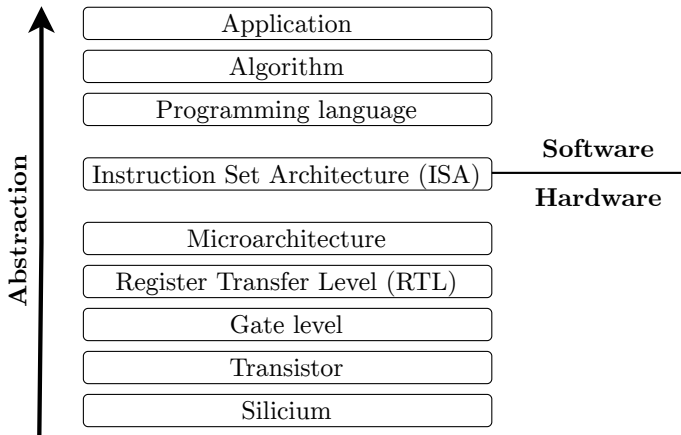
Abstraction hierarchy

“The essence of abstraction is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context.”

John V. Guttag

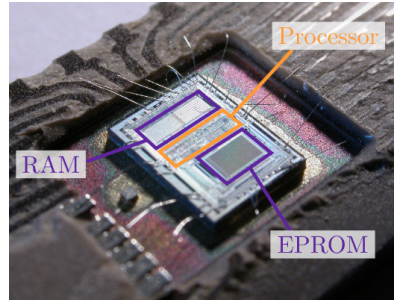
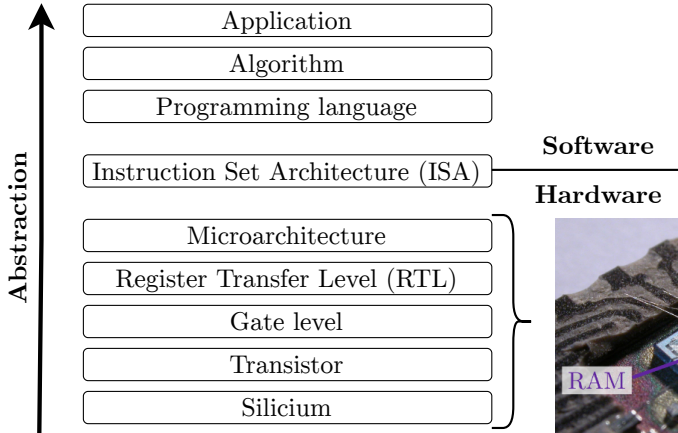
Context for execution integrity

Abstraction hierarchy



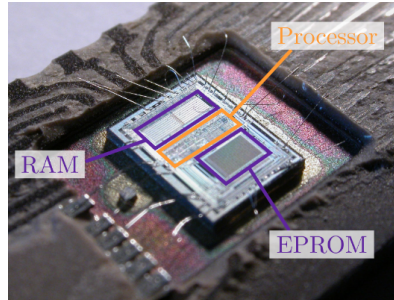
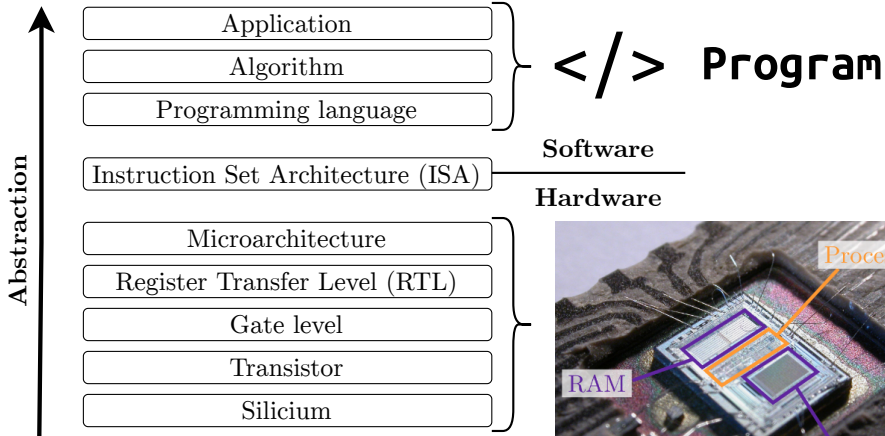
Context for execution integrity

Abstraction hierarchy



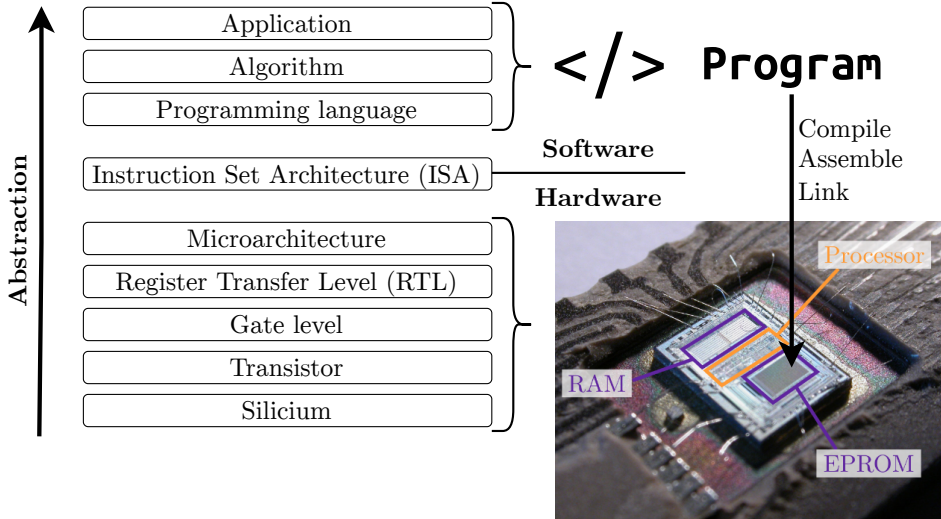
Context for execution integrity

Abstraction hierarchy



Context for execution integrity

Abstraction hierarchy

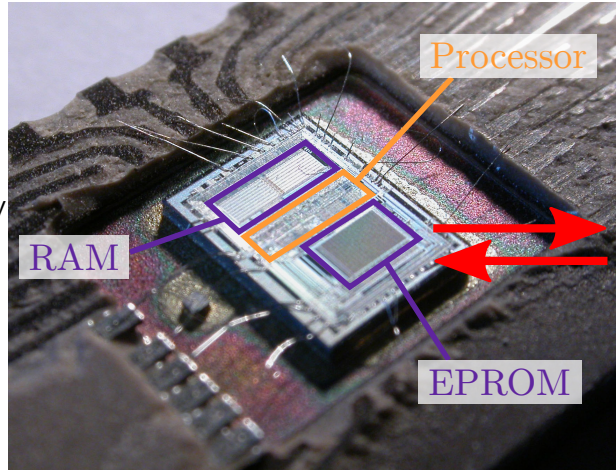


Context for execution integrity

Threat model

Threat model

- Read instructions in memory
- Write new instructions in memory



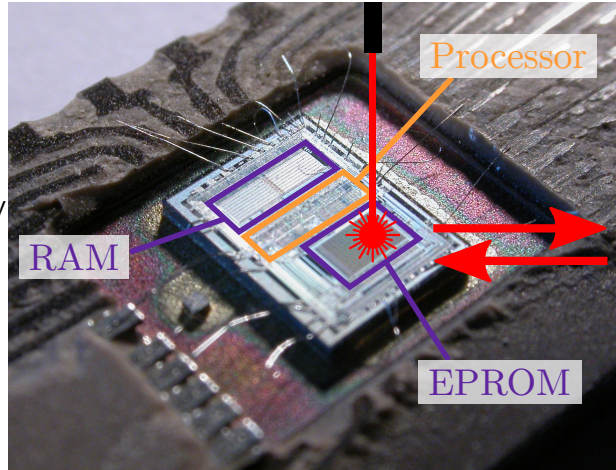
Context for execution integrity

Threat model

Threat model

- Read instructions in memory
- Write new instructions in memory
- FIA on instruction memory

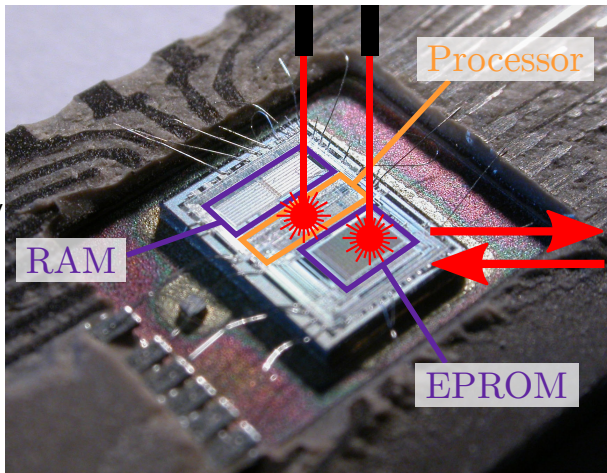
FIA: Fault Injection Attacks



Threat model

- Read instructions in memory
- Write new instructions in memory
- FIA on instruction memory
- FIA on processor control logic

FIA: Fault Injection Attacks



Context for execution integrity

Goal

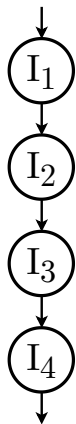
Security properties

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

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



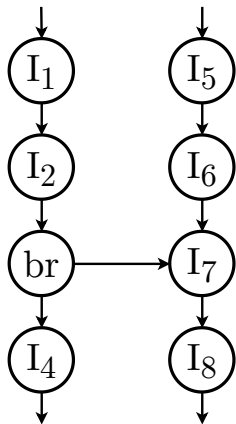
Instructions

- Sequential execution

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



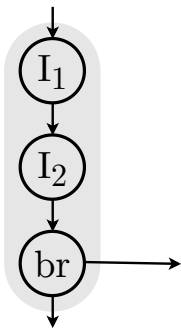
Instructions

- Sequential execution
- Non-sequential execution

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



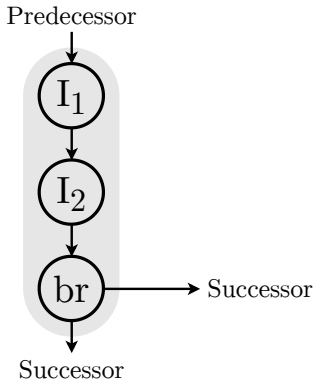
Basic Block:

- Sequential execution
- One entry point (first instruction)
- One exit point (last instruction)

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



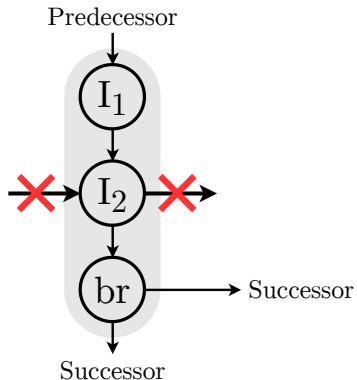
Basic Block:

- Sequential execution
- One entry point (first instruction)
- One exit point (last instruction)

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



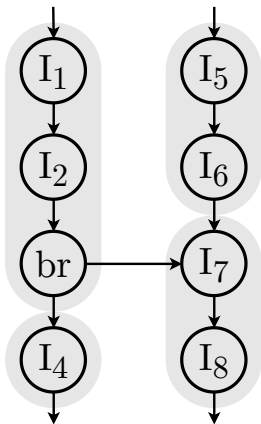
Basic Block:

- Sequential execution
- One entry point (first instruction)
- One exit point (last instruction)

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



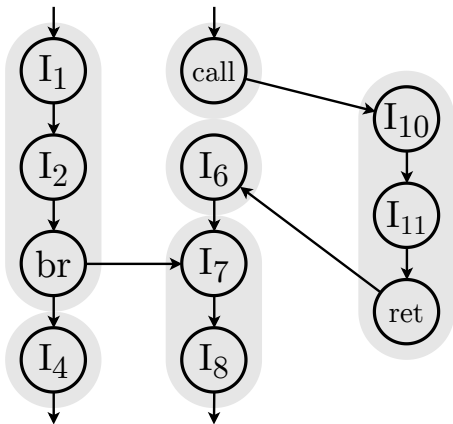
Basic Block:

- Sequential execution
- One entry point (first instruction)
- One exit point (last instruction)

Context for execution integrity

Control Flow and Instruction Integrity

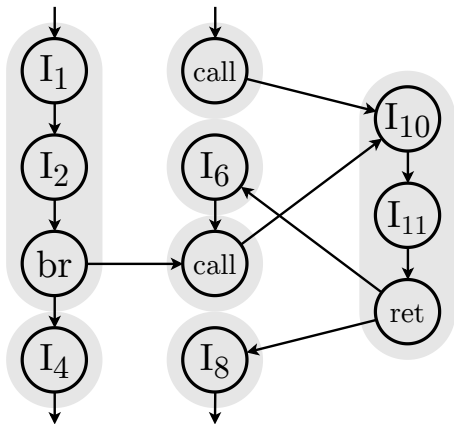
Program execution



Context for execution integrity

Control Flow and Instruction Integrity

Program execution

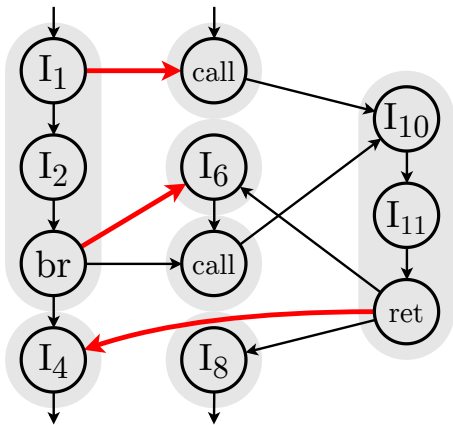


Control Flow Graph

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



Control Flow Graph

Control Flow Integrity

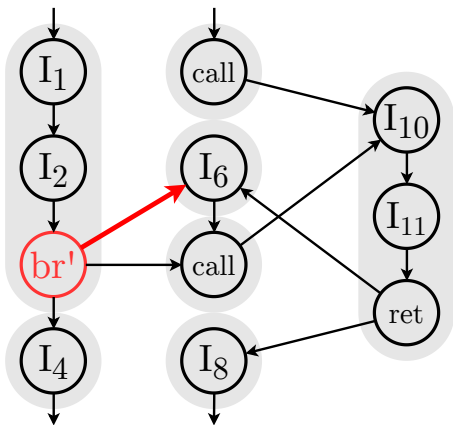
Tamper with:

- PC
- data
- microarchitecture

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



Control Flow Graph

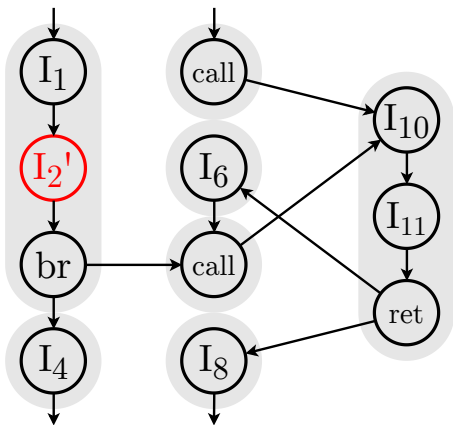
Control Flow Integrity

Instruction Integrity

Context for execution integrity

Control Flow and Instruction Integrity

Program execution



Control Flow Graph

Control Flow Integrity

Instruction Integrity

Context for execution integrity

Control Flow and Instruction Integrity

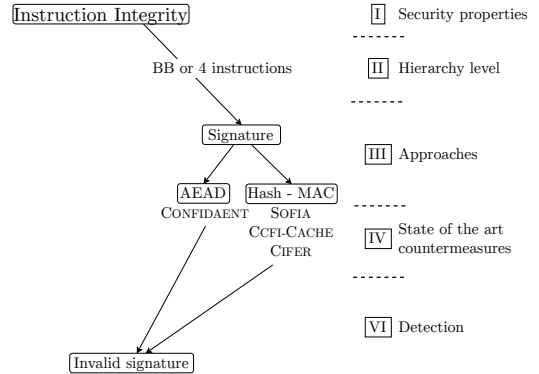
Bibliography

Context for execution integrity

Control Flow and Instruction Integrity

Bibliography

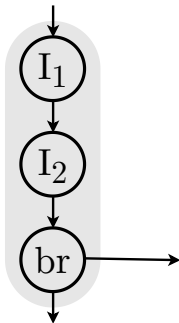
Mechanisms for Instruction Integrity



Context for execution integrity

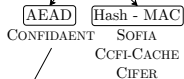
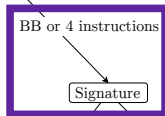
Control Flow and Instruction Integrity

Bibliography



Mechanisms for Instruction Integrity

Instruction Integrity



Invalid signature

I Security properties

II Hierarchy level

III Approaches

IV State of the art countermeasures

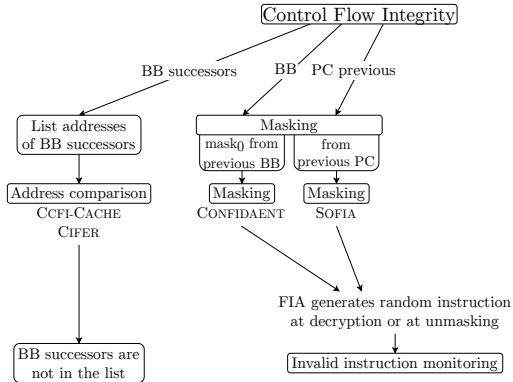
VI Detection

Context for execution integrity

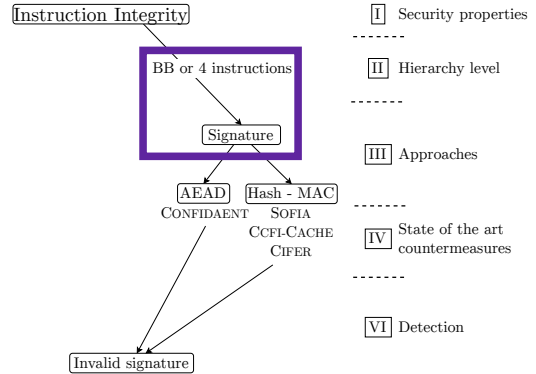
Control Flow and Instruction Integrity

Bibliography

Mechanisms for Control Flow Integrity



Mechanisms for Instruction Integrity



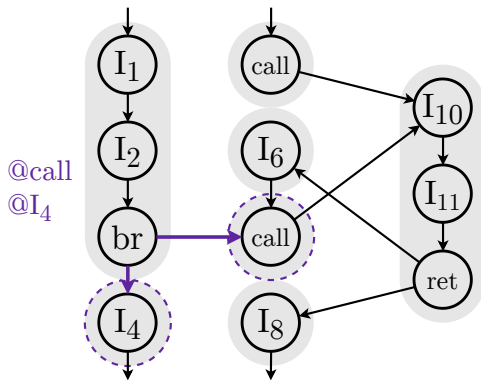
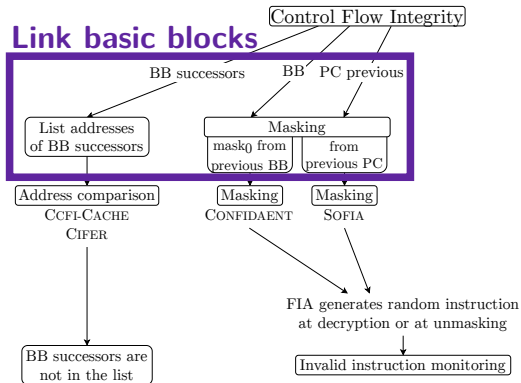
- I Security properties
-
- II Hierarchy level
-
- III Approaches
-
- IV State of the art countermeasures
-
- VI Detection

Context for execution integrity

Control Flow and Instruction Integrity

Bibliography

Mechanisms for Control Flow Integrity



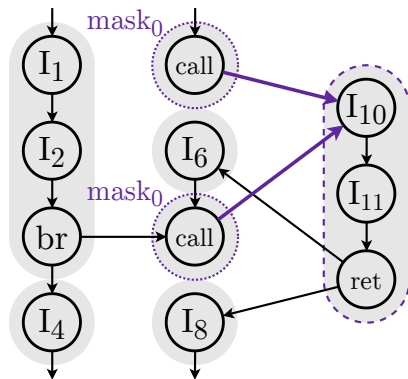
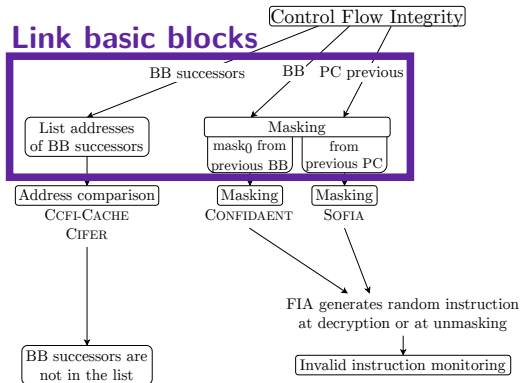
Control Flow Graph

Context for execution integrity

Control Flow and Instruction Integrity

Bibliography

Mechanisms for Control Flow Integrity



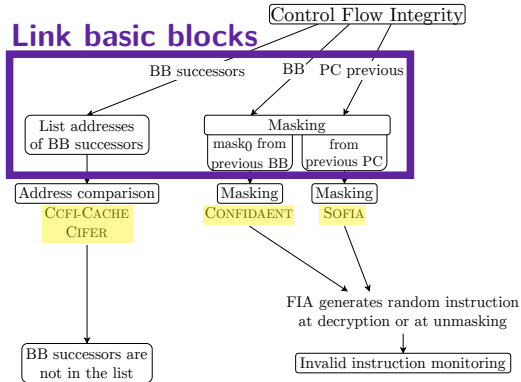
Control Flow Graph

Context for execution integrity

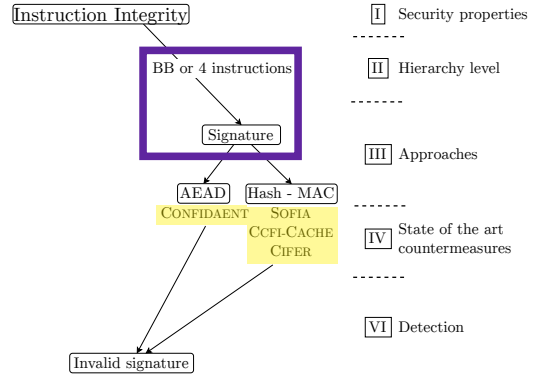
Control Flow and Instruction Integrity

Bibliography

Mechanisms for Control Flow Integrity



Mechanisms for Instruction Integrity

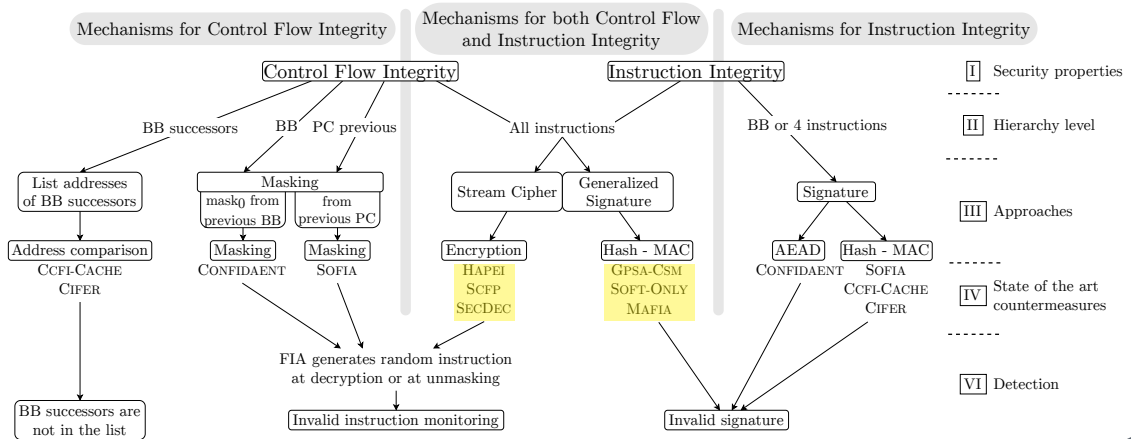


- I Security properties
-
- II Hierarchy level
-
- III Approaches
-
- IV State of the art countermeasures
-
- VI Detection

Context for execution integrity

Control Flow and Instruction Integrity

Bibliography

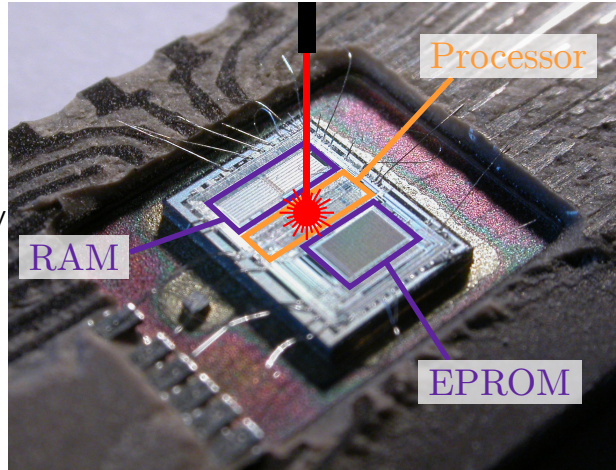


Context for execution integrity

Control Signal Integrity

Threat model

- Read instructions in memory
- Write new instructions in memory
- FIA in instruction memory
- FIA in processor control logic



FIA: Fault Injection Attacks

Context for execution integrity

Control Signal Integrity

Program execution in a pipeline

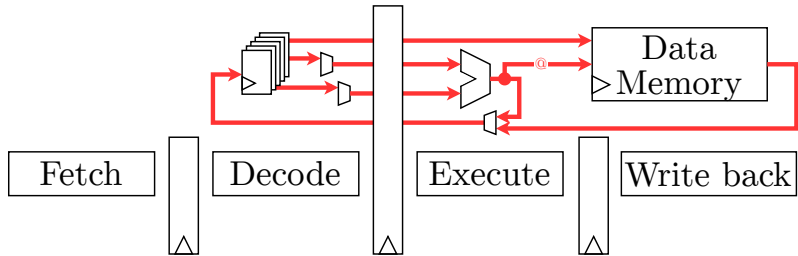


Context for execution integrity

Control Signal Integrity

Program execution in a pipeline

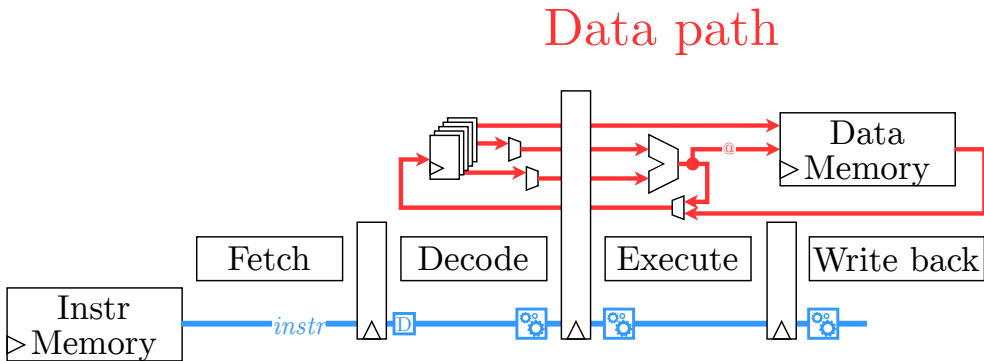
Data path



Context for execution integrity

Control Signal Integrity

Program execution in a pipeline



Data path

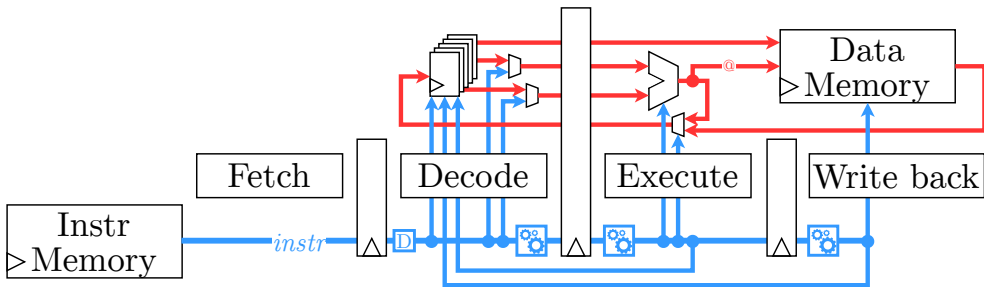
Control path

Context for execution integrity

Control Signal Integrity

Program execution in a pipeline

Data path



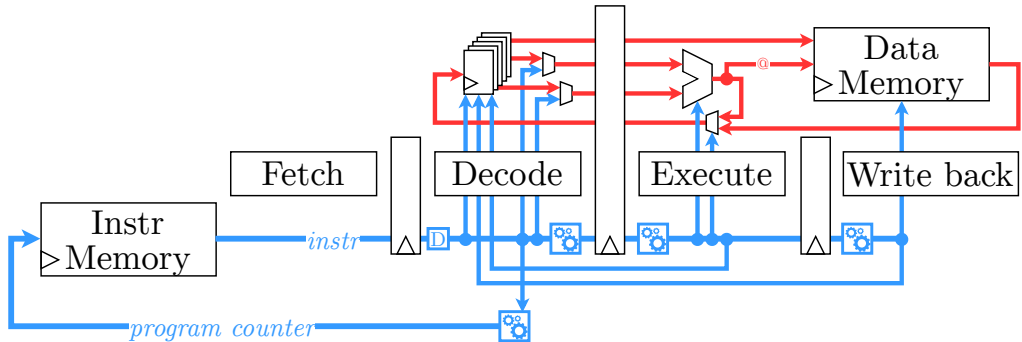
Control path

Context for execution integrity

Control Signal Integrity

Program execution in a pipeline

Data path

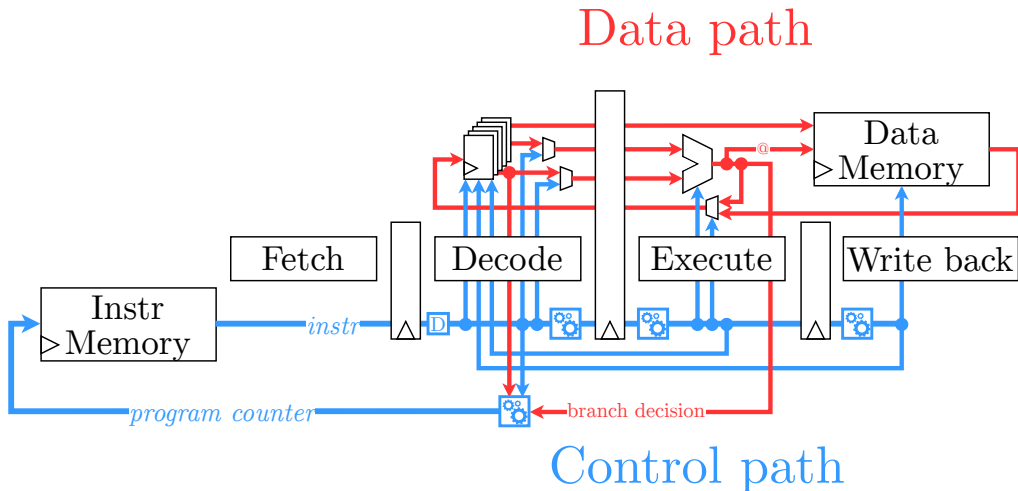


Control path

Context for execution integrity

Control Signal Integrity

Program execution in a pipeline



Context for execution integrity

Control Signal Integrity

Program execution in a pipeline

GPSA-CSM [11]

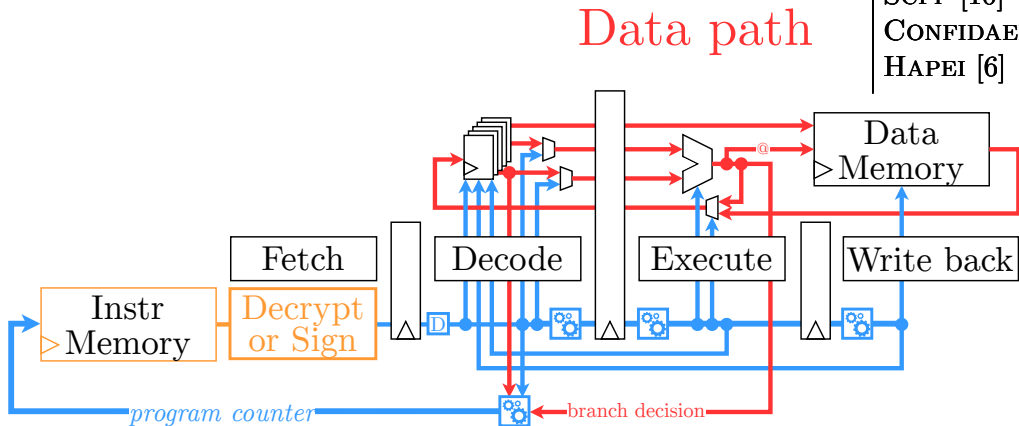
SOPIA [4]

CCFI-CACHE [3]

SCFP [10]

CONFIDAENT [9]

HAPEI [6]

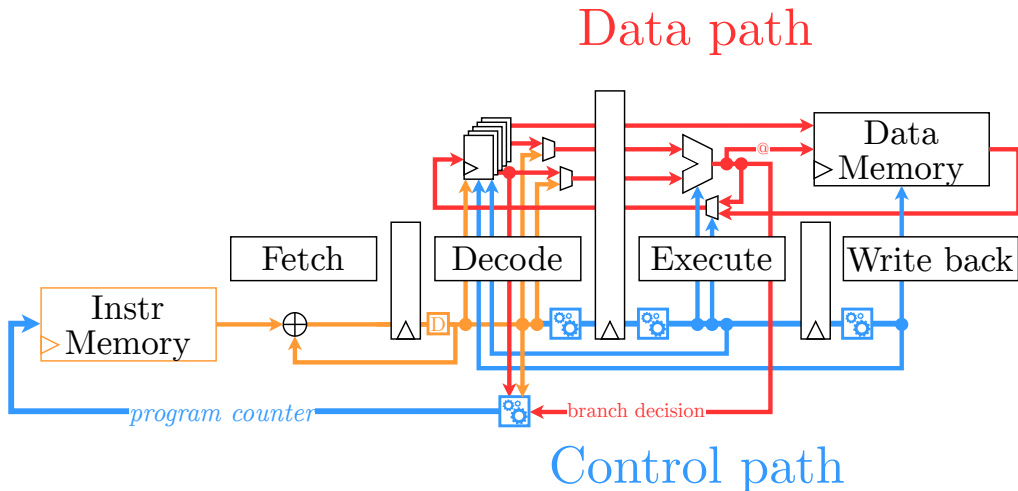


Control path

Context for execution integrity

Control Signal Integrity

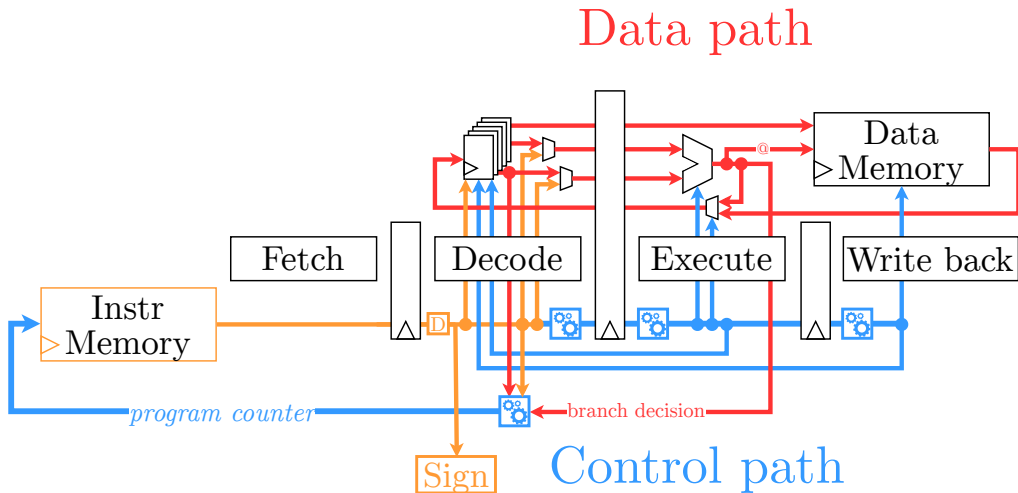
Program execution in a pipeline



Context for execution integrity

Control Signal Integrity

Program execution in a pipeline



Context for execution integrity

State-of-the-art execution integrity

Name	Instruction Confidentiality	Control Flow Integrity	Instruction Integrity	Control Signal Integrity	Fine-grained CFI	No clock cycle penalty	No memory overhead
INSTR. ENC. [5]	✓		✓			✓	(✓)
CONFIDAENT [9]	✓	✓	✓				
SOFIA [4]	✓	✓	✓		✓		
CCFI-CACHE [3]		✓	✓		✓	✓	
CIFER [13]		✓	✓	(✓)		✓	
SOFT-ONLY [1]		✓	✓				
SECDEC [8]		✓	✓	(✓)			
HAPFI [6]	✓	✓	✓		✓		
SCFP [10]	✓	✓	✓				
GPSA-CSM [11]		✓	✓				
MAFIA [2]		✓	✓	✓			
This work	✓	✓	✓	✓	✓	✓	✓

Context for execution integrity

State-of-the-art execution integrity

Name		Control Flow Integrity	Instruction Integrity	Control Signal Integrity		
INSTR. ENC. [5]	✓		✓		✓	(✓)
CONFIDAENT [9]	✓	✓	✓			
SOFIA [4]	✓	✓	✓		✓	
CCFI-CACHE [3]		✓	✓		✓	✓
CIFER [13]		✓	✓	(✓)		✓
SOFT-ONLY [1]		✓	✓			
SECDEC [8]		✓	✓	(✓)		
HAPPI [6]	✓	✓	✓		✓	
SCFP [10]	✓	✓	✓			
GPSA-CSM [11]		✓	✓			
MAFIA [2]		✓	✓	✓		
This work	✓	✓	✓	✓	✓	✓

Annotations:

- A purple box highlights the 'Control Signal Integrity' column for the rows: INSTR. ENC. [5], CONFIDAENT [9], SOFT-ONLY [1], SECDEC [8], and MAFIA [2].
- Two purple arrows point from the text "Only signals in decode" to the highlighted cells for CONFIDAENT [9] and SECDEC [8].
- A purple arrow points from the text "Control signal redundancy" to the highlighted cell for MAFIA [2].

Proposed scheme

- 1 Who am I?
- 2 Context for execution integrity
- 3 Proposed scheme**
 - Impact on the state of the Art
 - Our mechanism
 - Implementation
 - Need for patches
 - Patch policy
 - Solution flow
 - Software model of control signal propagation
 - cv32e40p deterministic control signals
- 4 Validation and characterization
- 5 Conclusion

Security properties

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

Proposed scheme

Security properties

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

Constraints

- Zero clock cycle penalty
- No inserted instruction
- No recompilation needed

Proposed scheme

Security properties

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

Constraints

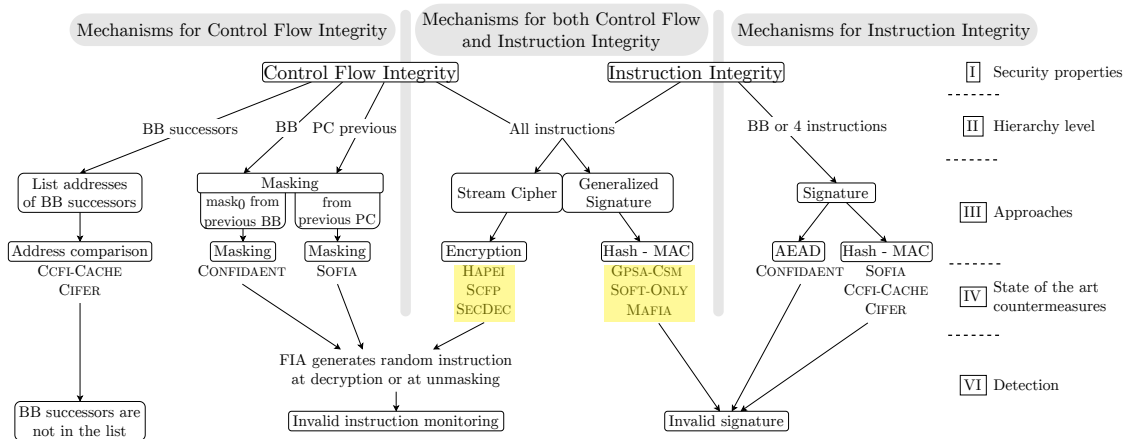
- Zero clock cycle penalty
- No inserted instruction
- No recompilation needed

Assumptions (microcontroller):

- No memory cache
- Cycle-per-instruction does not depend on data
- Indirect jump destinations are considered known

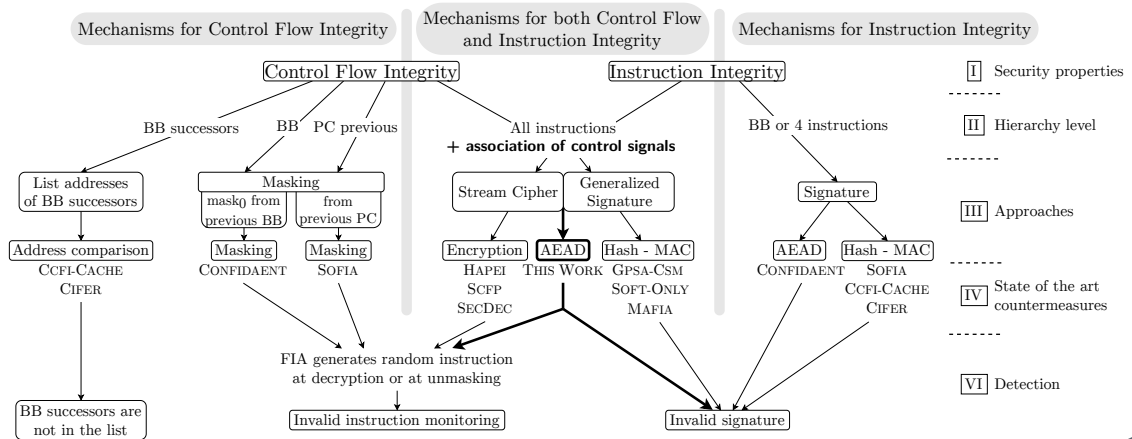
Proposed scheme

Impact on the state of the Art



Proposed scheme

Impact on the state of the Art



Chained Instruction Encryption with Associated Control Signals

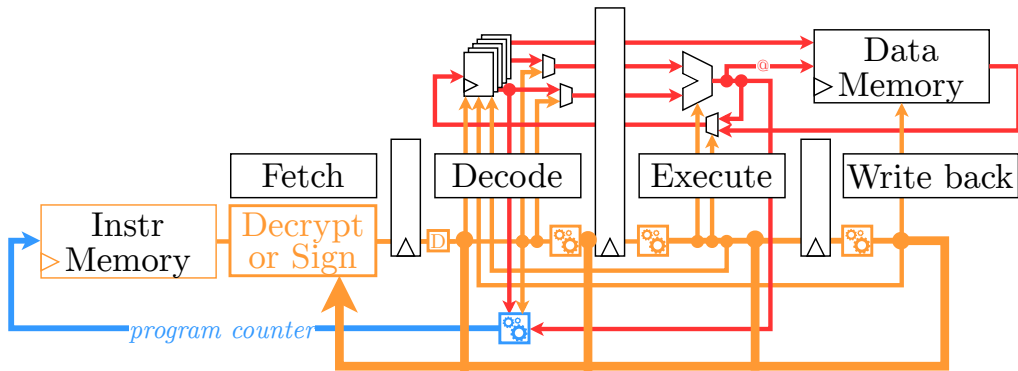
Chained Instruction Encryption with Associated Control Signals

- ✓ **Confidentiality** of Instructions
- ✓ **Integrity** of Control Flow
- ✓ **Integrity** of Instructions
- ✓ **Integrity** of Control Signals

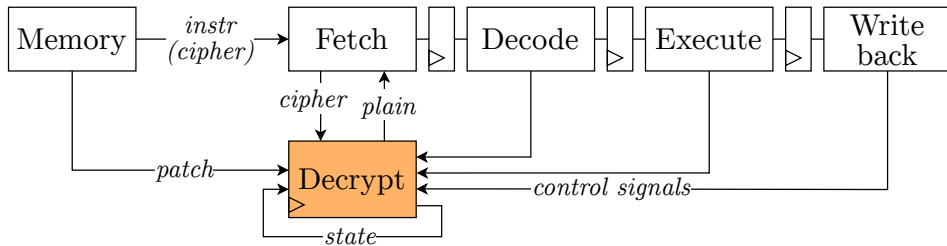
Proposed scheme

Our mecanism

Data path

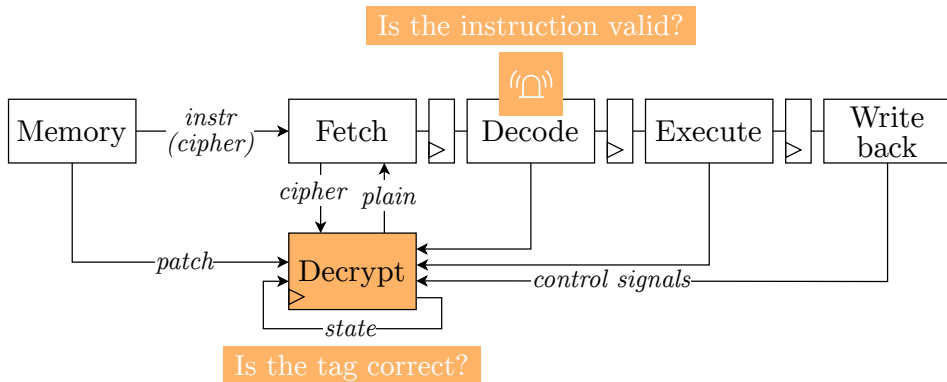


Chained Instruction Encryption with Associated Control Signals



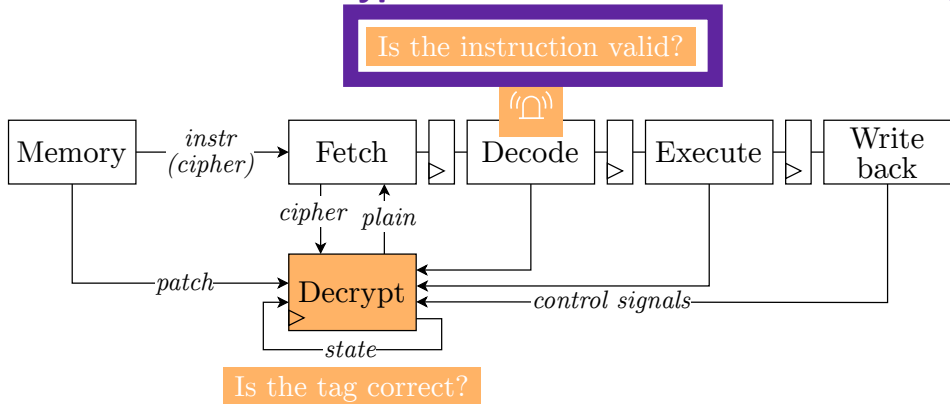
- 1 Chained Encryption of Instructions (before programming memory)
- 2 On-the-fly Decryption

Chained Instruction Encryption with Associated Control Signals



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

Chained Instruction Encryption with Associated Control Signals



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

Proposed scheme Implementation

RISC-V core: cv32e40p

CV32E40P: RISC-V core

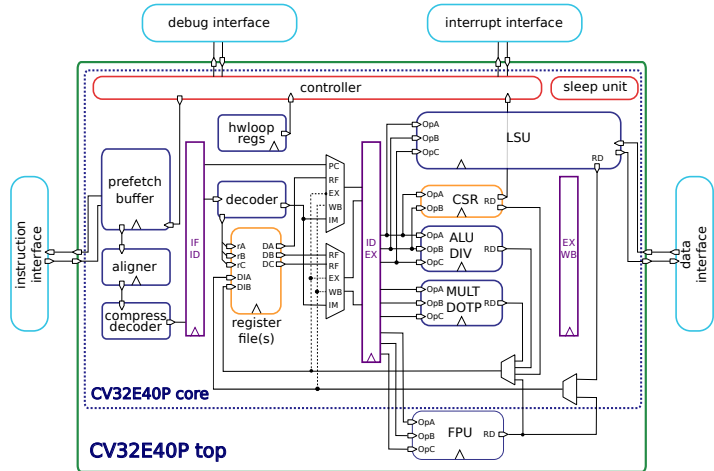
Proposed scheme

Implementation

RISC-V core: cv32e40p

CV32E40P: RISC-V core

- Embedded system
- 32-bit
- 4-stage
- In-order
- Forwarding



Proposed scheme

Implementation

Authenticated Encryption: ASCON

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

Proposed scheme

Implementation

Authenticated Encryption: ASCON

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

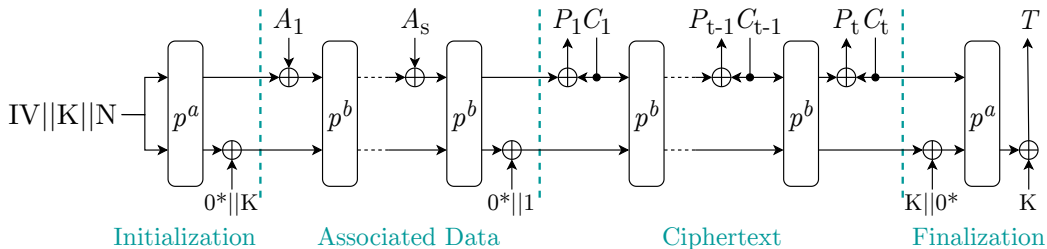
- Winner of CAESAR Authenticated Encryption - Lightweight Cryptography (2019)
- Winner of NIST - Lightweight Cryptography (2023) ⇒ **standardize the ASCON family**
- Lightweight (better Throughput per Area than AES [1])
- Highly tested
- Large security margins

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

Proposed scheme

Implementation

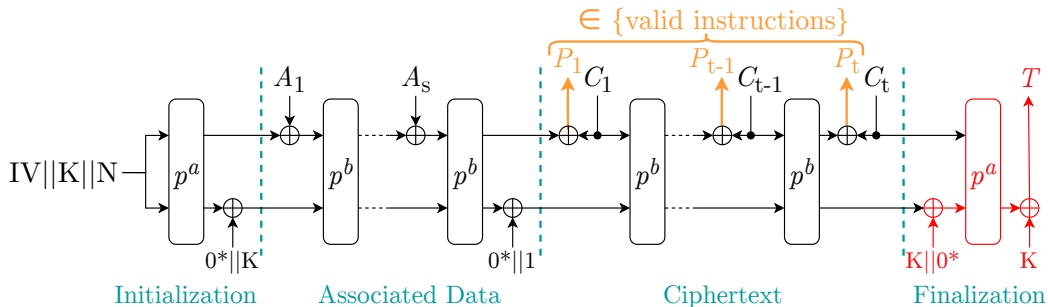
Authenticated Encryption: ASCON



Proposed scheme

Implementation

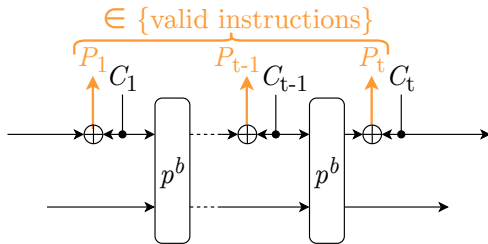
Authenticated Encryption: ASCON



Proposed scheme

Implementation

Authenticated Encryption: ASCON



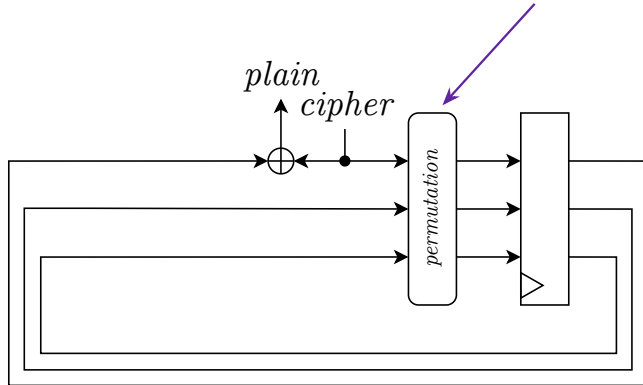
Ciphertext

Proposed scheme

Implementation

Authenticated Encryption: ASCON

6 permutations

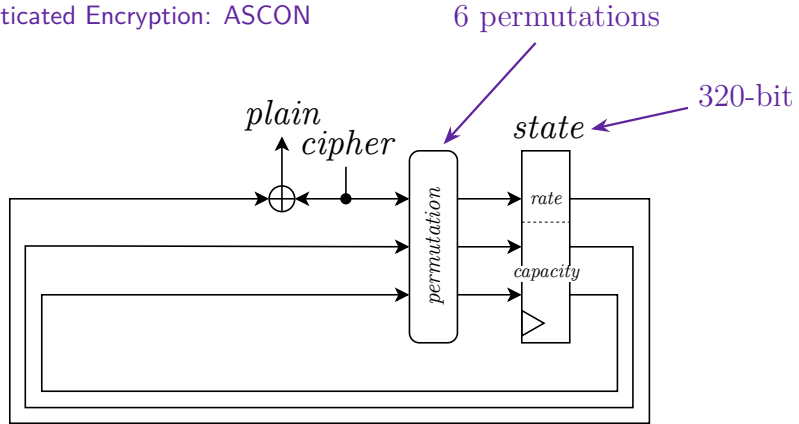


Hardware decryption and Software encryption (symmetric)

Proposed scheme

Implementation

Authenticated Encryption: ASCON

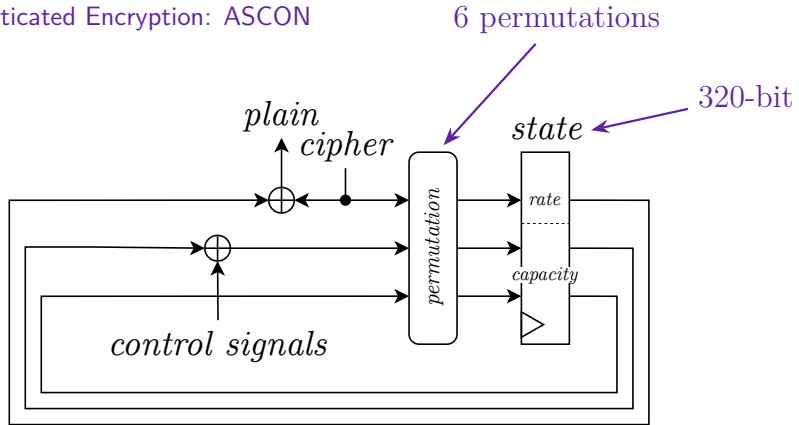


Hardware decryption and Software encryption (symmetric)

Proposed scheme

Implementation

Authenticated Encryption: ASCON

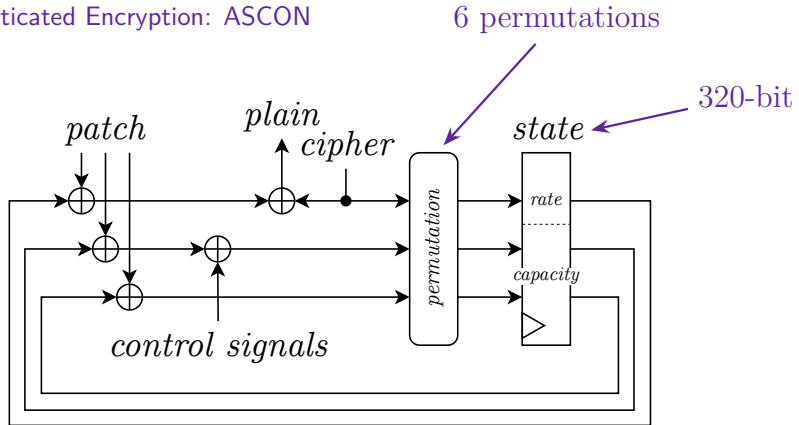


Hardware decryption and Software encryption (symmetric)

Proposed scheme

Implementation

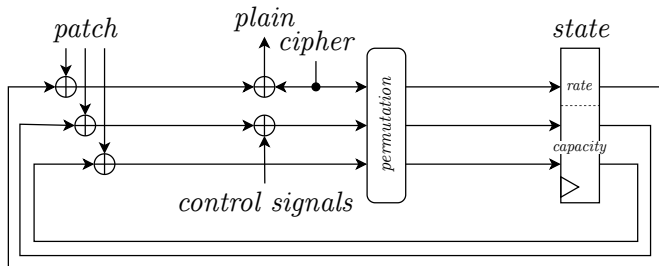
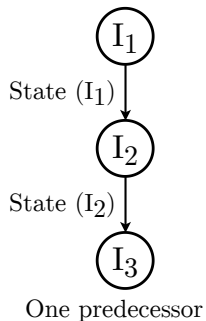
Authenticated Encryption: ASCON



Hardware decryption and Software encryption (symmetric)

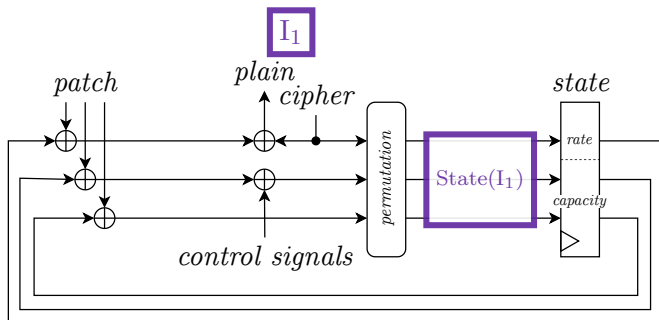
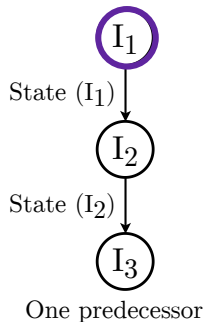
Proposed scheme

Need for patches



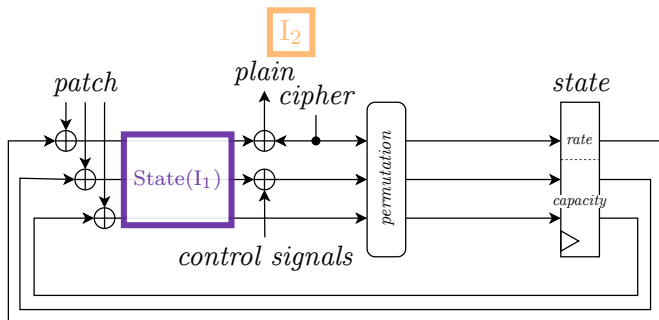
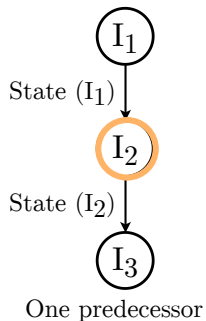
Proposed scheme

Need for patches



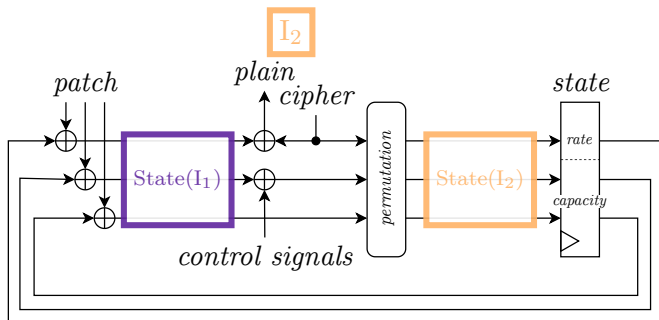
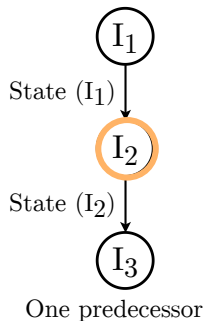
Proposed scheme

Need for patches



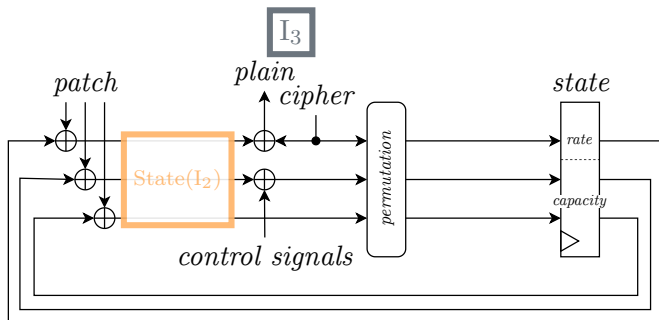
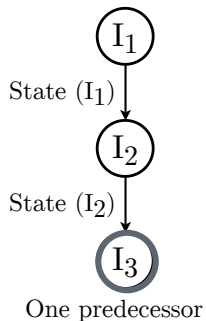
Proposed scheme

Need for patches



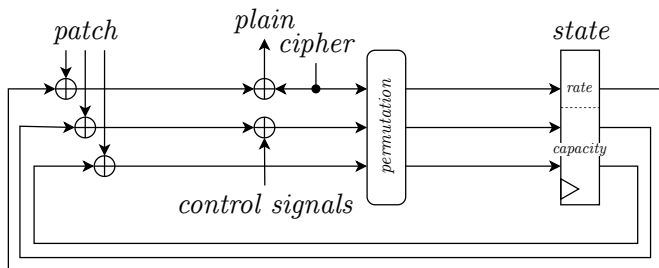
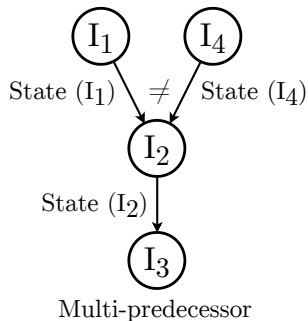
Proposed scheme

Need for patches



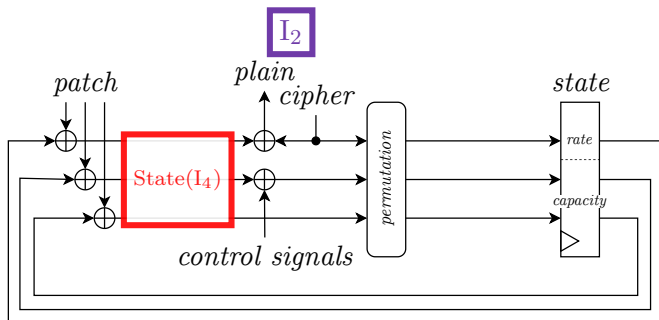
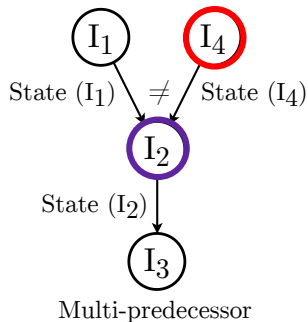
Proposed scheme

Need for patches



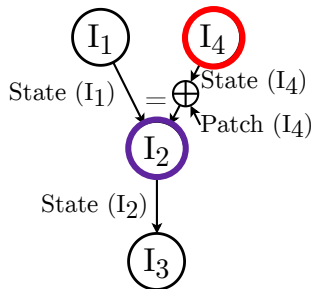
Proposed scheme

Need for patches



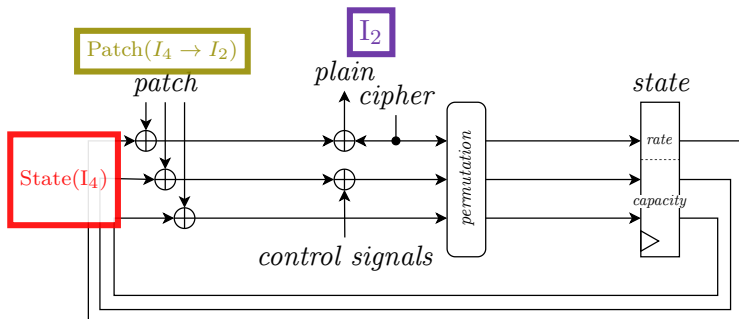
Proposed scheme

Need for patches



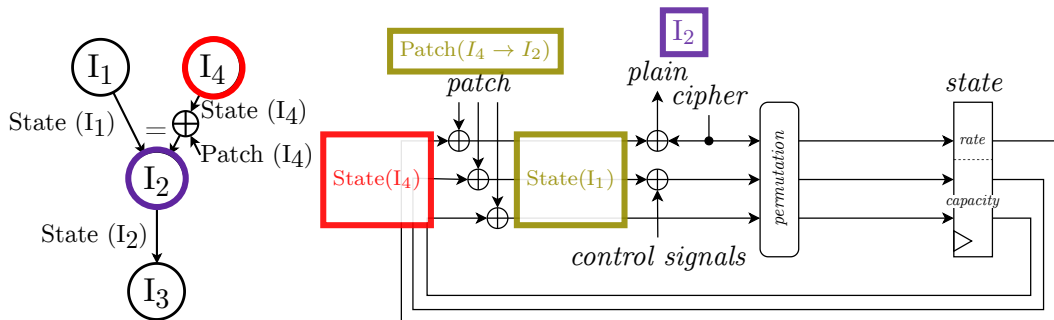
Patching multi-predecessor

Patch (I_4) = State (I_1) xor State (I_4)



Proposed scheme

Need for patches

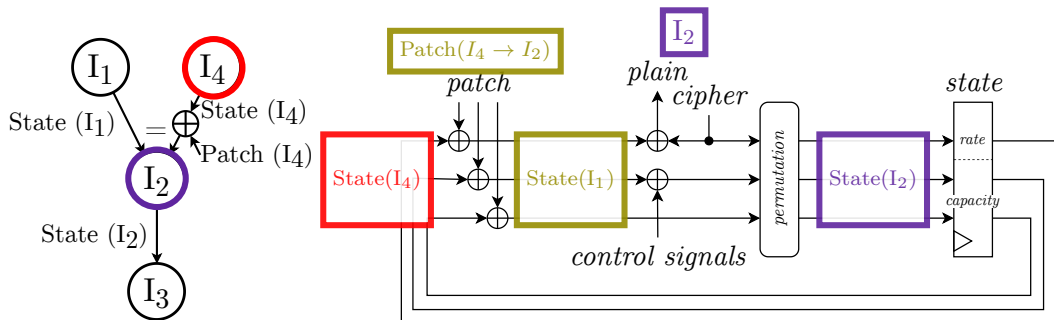


Patching multi-predecessor

Patch (I_4) = State (I_1) xor State (I_4)

Proposed scheme

Need for patches



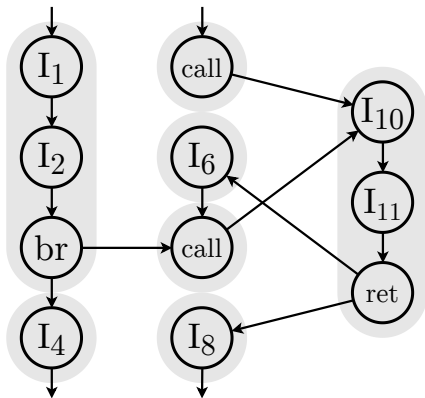
Patching multi-predecessor

Patch (I_4) = State (I_1) xor State (I_4)

Proposed scheme

Patch policy

“Sequential encryption for sequential instructions”

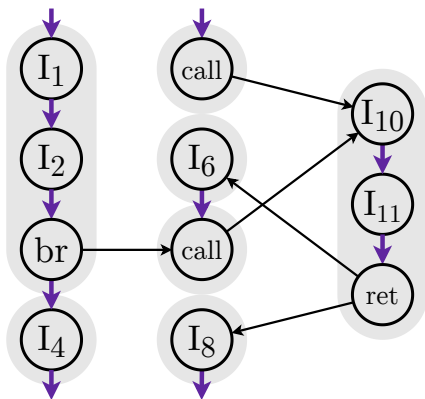


Control Flow Graph

Proposed scheme

Patch policy

“Sequential encryption for sequential instructions”



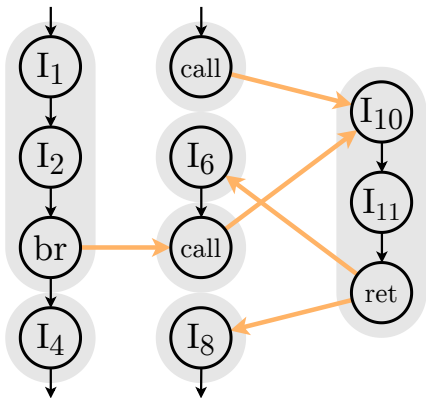
Chain sequential instr.

Control Flow Graph

Proposed scheme

Patch policy

“Sequential encryption for sequential instructions”



Patches

(for non-sequential control transfer)

Control Flow Graph

Proposed scheme

Patch policy

- Patches stored in external memory
- Patches addressed by Program Counter
 - No need for custom instructions

⇒ **zero clock cycle overhead**

Proposed scheme

Patch policy

Patch allocation

Table: Control transfer instruction characteristics in RISC-V.

Instruction	Pseudoinstruction	Successor addresses
Direct Jump	<i>jal rd, imm</i>	$pc + imm$
Conditional Branch	<i>b rs1, rs2, imm</i>	$pc + 4 / pc + imm$
Indirect jump	<i>jalr rd, rs1, imm</i>	$rs1 + imm$

Proposed scheme

Patch policy

Patch allocation

Table: Patch address of Control transfer instructions.

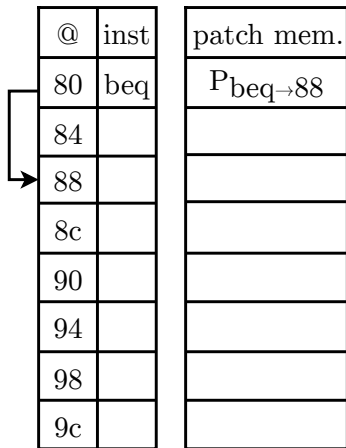
Control transfer instruction	Patch address
Direct Jump	$@_{jal} = P_{jal \rightarrow dest}$
Conditional Branch Taken	$@_{br} = P_{br \rightarrow dest}$
Conditional Branch Not Taken	<i>no need for patch</i>
Indirect jump	$@_{dest} = P_{jalr \rightarrow dest}$
Indirect jump (patch reallocated)	$@_{free} = P_{jalr \rightarrow dest}$

Proposed scheme

Patch policy

Patch allocation

@	inst	patch mem.
80	beq	$P_{\text{beq} \rightarrow 88}$
84		
88		
8c		
90		
94		
98		
9c		

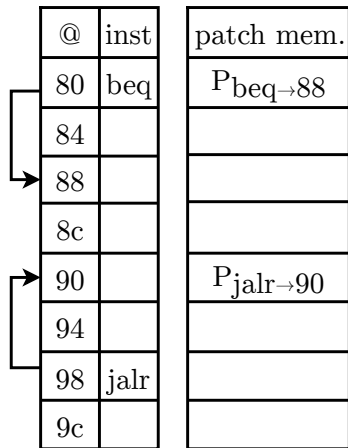


(a) Patch allocation

Proposed scheme

Patch policy

Patch allocation

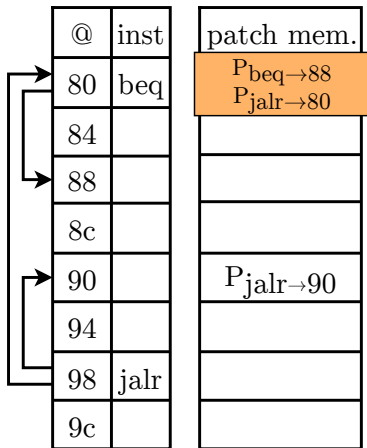


(a) Conflict at address 80

Proposed scheme

Patch policy

Patch allocation

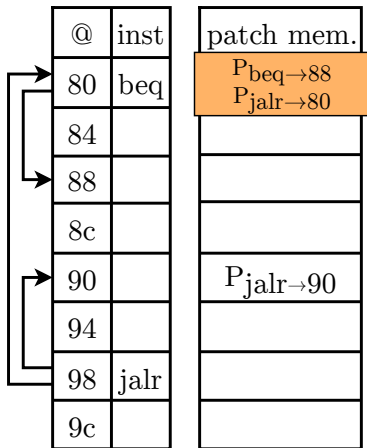


(a) Conflict at address 80

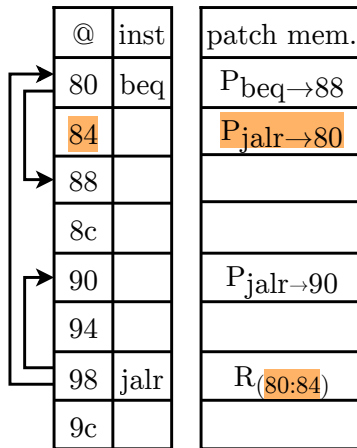
Proposed scheme

Patch policy

Patch allocation



(a) Conflict at address 80

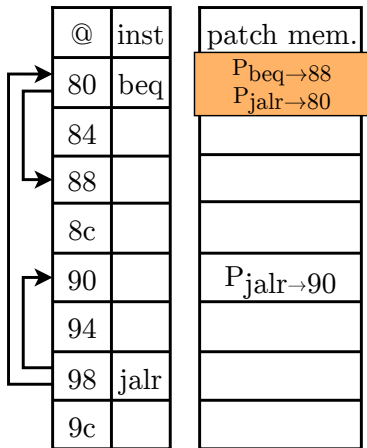


(b) Reallocation for jalr patches

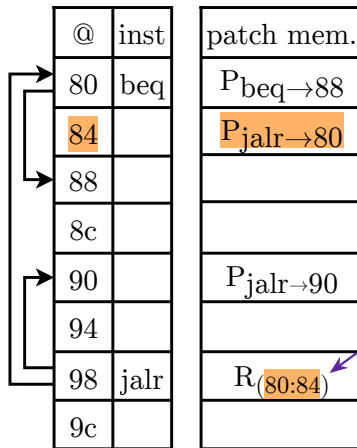
Proposed scheme

Patch policy

Patch allocation



(a) Conflict at address 80



Limitation
12 successors

(b) Reallocation for jalr patches

Proposed scheme

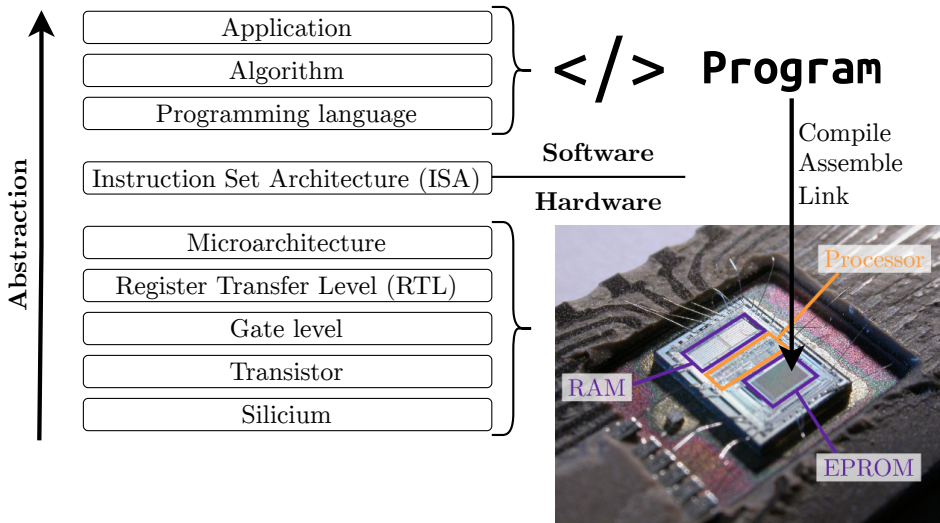
Solution flow

Soft(no need for recompilation):

Hard:

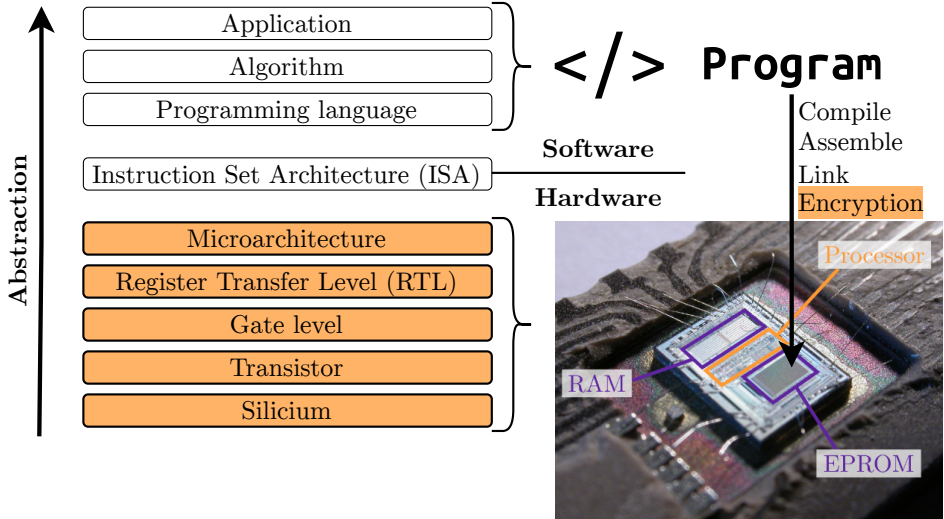
Proposed scheme

Solution flow



Proposed scheme

Solution flow



Proposed scheme

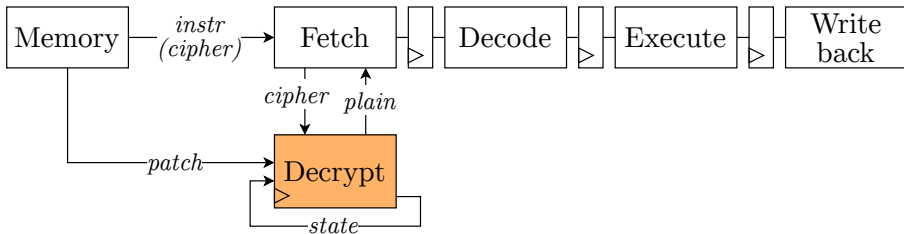
Solution flow

Soft(no need for recompilation):

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

Hard:

- 1 Patch memory
- 2 ASCON decryption (+ FSM)



Proposed scheme

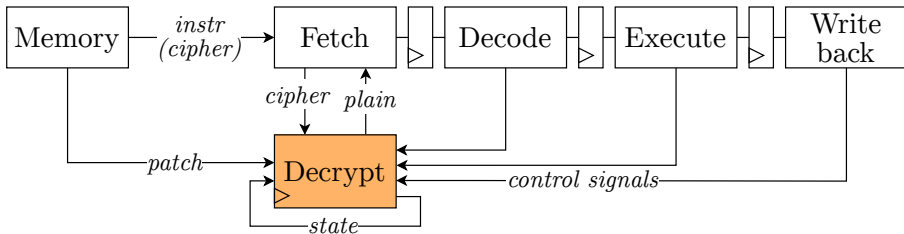
Solution flow

Soft(no need for recompilation):

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

Hard:

- 1 Patch memory
- 2 ASCON decryption (+ FSM)
- 3 Control Signal routing



Proposed scheme

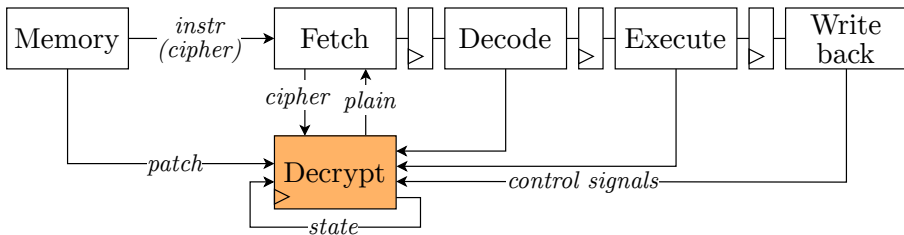
Solution flow

Soft(no need for recompilation):

- 1 Encrypt instructions sequentially ✘
- 2 Build CFG **Control signal values?**
- 3 Generate patches ✘

Hard:

- 1 Patch memory
- 2 ASCON decryption (+ FSM)
- 3 Control Signal routing



Proposed scheme

Software model of control signal propagation

- ✓ Instructions (machine code)
- ✓ Microarchitecture netlist

→ Infer control signal values

Proposed scheme

Software model of control signal propagation

- ✓ Instructions (machine code)
- ✓ Microarchitecture netlist

→ Infer control signal values

Software model of control signal propagation

Proposed scheme

Software model of control signal propagation

- ✓ Instructions (machine code) → Infer control signal values
- ✓ Microarchitecture netlist

Software model of control signal propagation

- Encrypt and model signals for sequential executions
- Generate patches

Proposed scheme

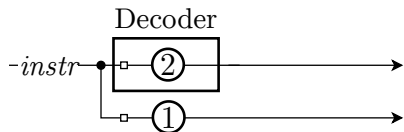
Software model of control signal propagation
4-stage pipeline

-instr—

Decode

Proposed scheme

Software model of control signal propagation
4-stage pipeline



Decode

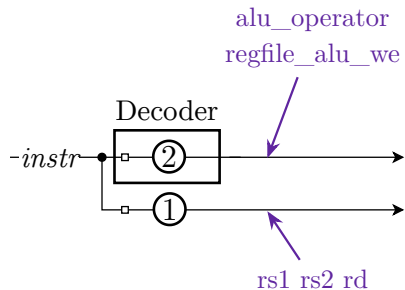
① Extracted

② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation
4-stage pipeline



Decode

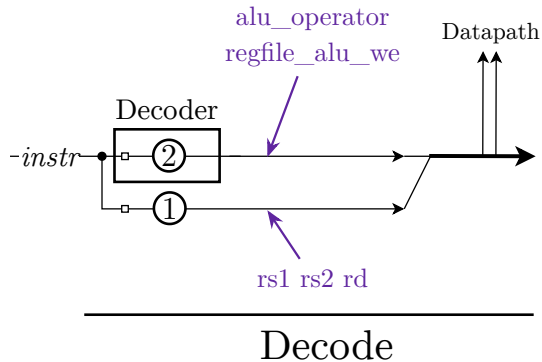
① Extracted

② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation
4-stage pipeline



① Extracted

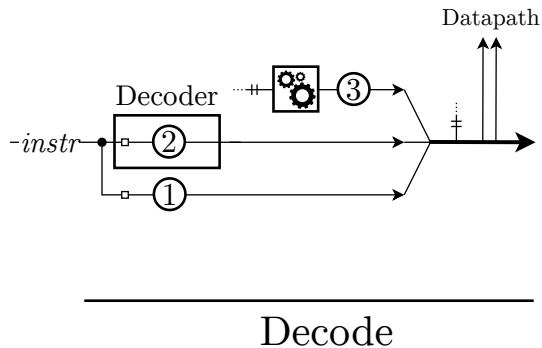
② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline



① Extracted

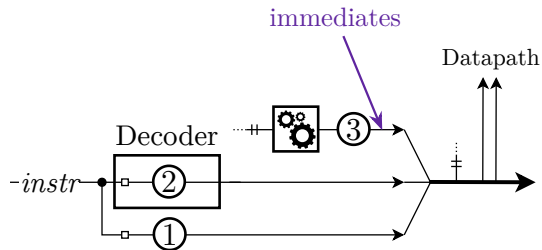
② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline



Decode

① Extracted

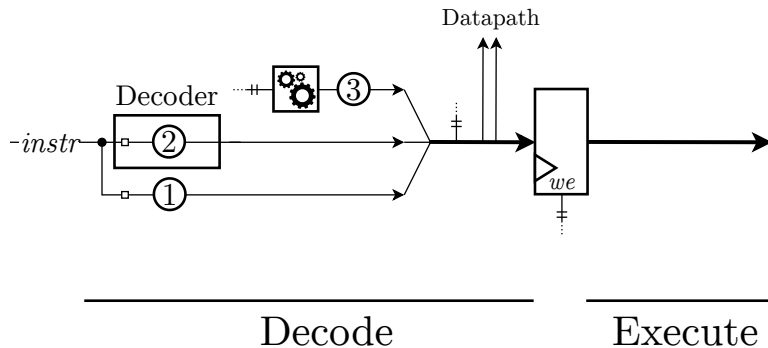
② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline



① Extracted

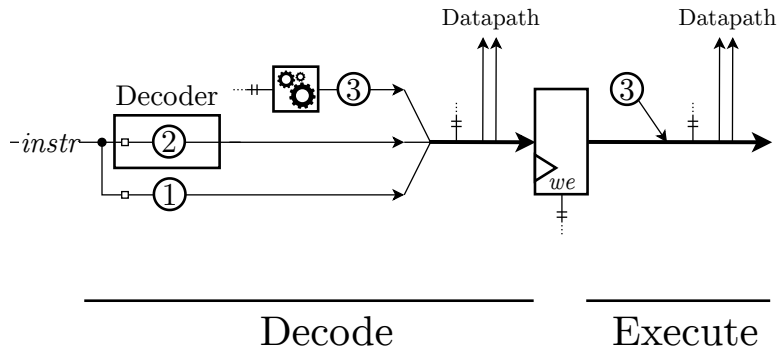
② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline



① Extracted

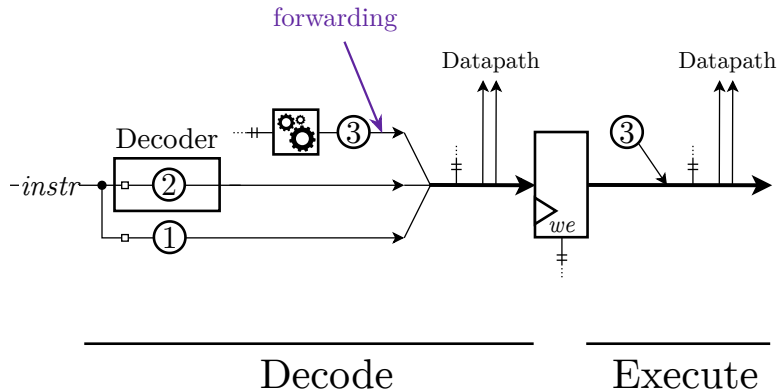
② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline



① Extracted

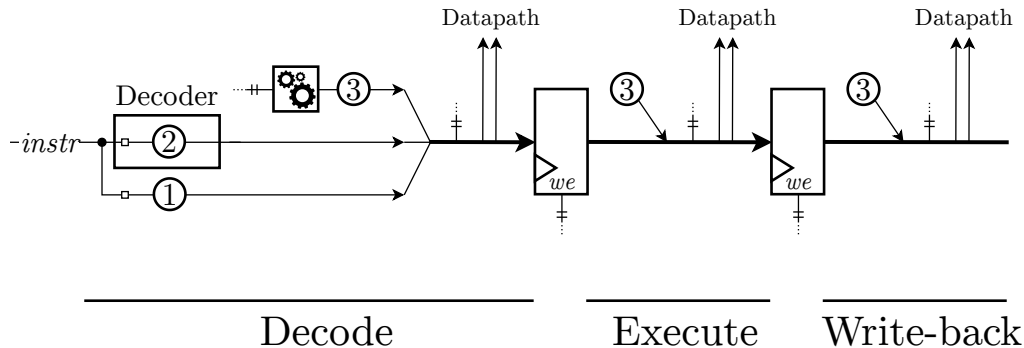
② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline



① Extracted

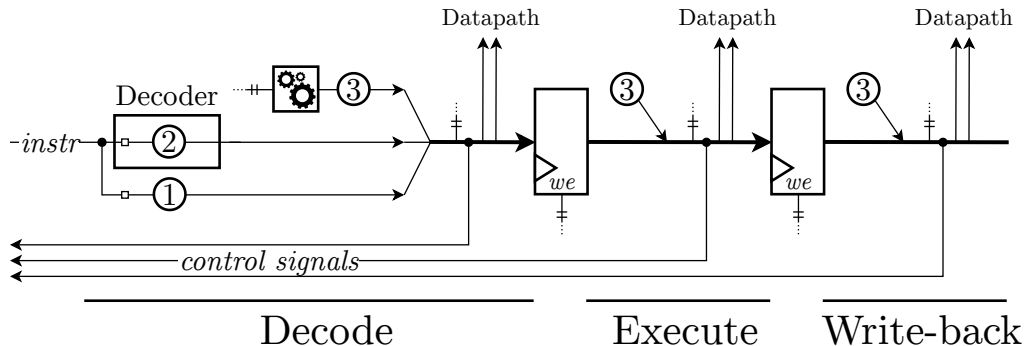
② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline



① Extracted

② Decoded

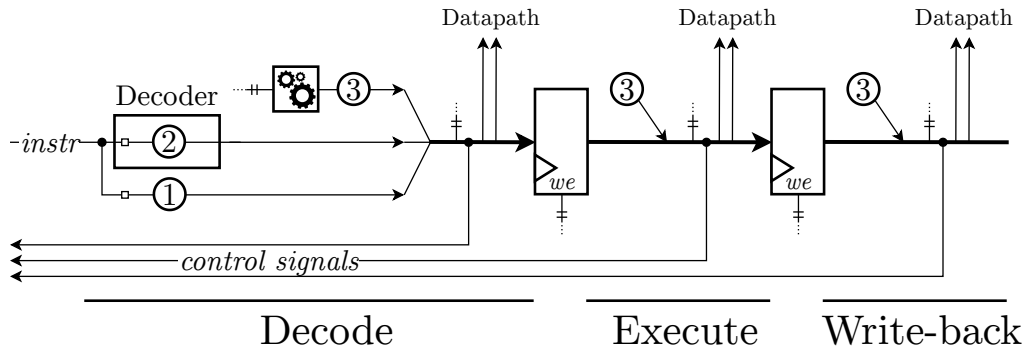
③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline

Deterministic control signals: depend only on instruction **S** (not data)



① Extracted

② Decoded

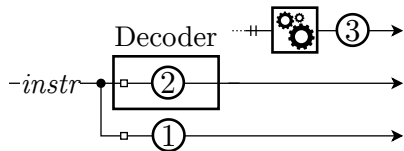
③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline

Deterministic control signals: depend only on instruction **S** (not data)



Decode

① Extracted

② Decoded

③ Combinational

Proposed scheme

Software model of control signal propagation

4-stage pipeline

→ Encrypt and model signals for sequential executions

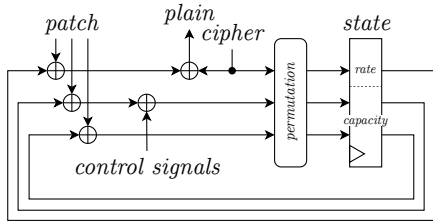
cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	

| addresses
| alu_operator

Proposed scheme

Software model of control signal propagation 4-stage pipeline

→ Encrypt and model signals for sequential executions



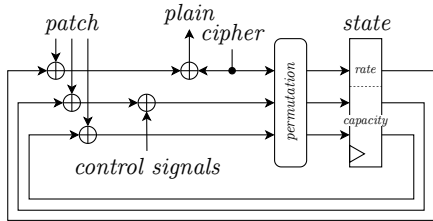
cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	

addresses | alu_operator

Proposed scheme

Software model of control signal propagation 4-stage pipeline

→ Encrypt and model signals for sequential executions



cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	

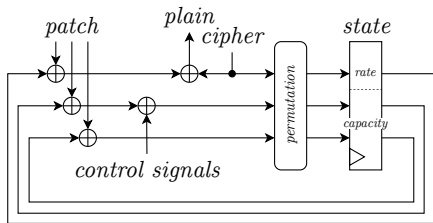
addresses | alu_operator

Proposed scheme

Software model of control signal propagation

4-stage pipeline

→ Encrypt and model signals for sequential executions



cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	

addresses | alu_operator

- ✓ Stall due to multi-cycle instructions
- ✓ Stall due to data dependancies

Proposed scheme

Software model of control signal propagation

Control transfer: adapt patches

→ Generate patches

Proposed scheme

Software model of control signal propagation

Control transfer: adapt patches

→ Generate patches

cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	
4	83C	838	18	18	
5	840	83C	0D	18	
6	82C	840	18	0D	
7	830	82C	18	03	
8	834	830	18	03	

| addresses | alu_operator |

branch fetched

branch decoded

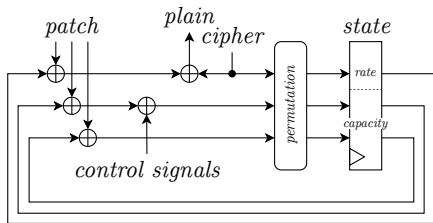
branch taken

Proposed scheme

Software model of control signal propagation

Control transfer: adapt patches

→ Generate patches



cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	
4	83C	838	18	18	
5	840	83C	0D	18	
6	82C	840	18	0D	
7	830	82C	18	03	
8	834	830	18	03	

branch fetched

branch decoded

branch taken

Patch

addresses

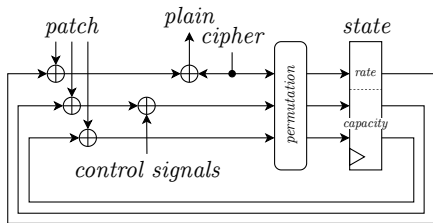
alu_operator

Proposed scheme

Software model of control signal propagation

Control transfer: adapt patches

→ Generate patches



✓ Stall/flush due to *jump*, *branch* execution

cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	
4	83C	838	18	18	
5	840	83C	0D	18	
6	82C	840	18	0D	
7	830	82C	18	03	
8	834	830	18	03	

| addresses | alu_operator |

branch fetched

branch decoded

branch taken

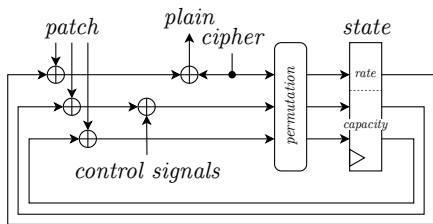
Patch

Proposed scheme

Software model of control signal propagation

Control transfer: adapt patches

→ Generate patches



✓ Stall/flush due to *jump, branch* execution

cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	
4	83C	838	18	18	
5	840	83C	0D	18	
6	82C	840	18	0D	
7	830	82C	18	03	
8	834	830	18	03	

addresses | alu_operator | Patch (extra bits)

branch fetched

branch decoded

branch taken

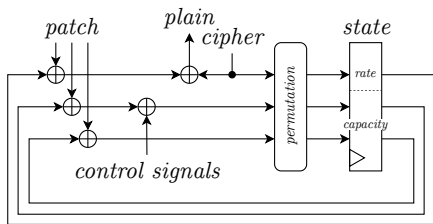
Patch
(extra bits)

Proposed scheme

Software model of control signal propagation

Control transfer: adapt patches

→ Generate patches



✓ Stall/flush due to *jump*, *branch* execution

cyc	MEM	IF	ID	EX	WB
0	82C	828	18	03	
1	830	82C	0C	18	
2	834	830	18	0C	
3	838	834	18	18	
4	83C	838	18	18	
5	840	83C	0D	18	
6	82C	840	18	0D	
7	830	82C	18	03	
8	834	830	18	03	

branch fetched

branch decoded

branch taken

addresses | alu_operator

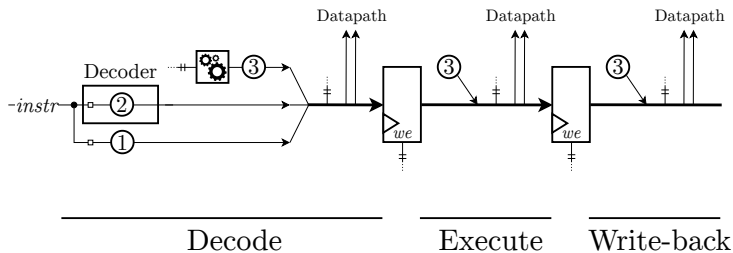
Patch
(extra bits)

Proposed scheme

cv32e40p deterministic control signals

Table: Deterministic Control Signals supported.

Origin	Decode	Execute	Write-back	examples
Extracted ^①	25 (47 bits)	12 (23 bits)	4 (6 bits)	<i>alu_operator</i>
Decoded ^②	4 (24 bits)	2 (12 bits)	1 (6 bits)	<i>rs1, rs2, rd</i>
Combinational ^③	2 (64 bits)	1 (1 bits)		<i>imm</i>



Validation and characterization

- 1 Who am I?
- 2 Context for execution integrity
- 3 Proposed scheme
- 4 **Validation and characterization**
 - core-v-verif-fpga environnement
 - Cycle-accurate simulations
 - FPGA
 - FPGA: Cross domain clocking
 - FPGA
 - Experimental validation
 - Probability of no-detection
- 5 Conclusion

Makefile based:

- Compilation (*Embench*) and software encryption flow

Makefile based:

- Compilation (*Embench*) and software encryption flow
- **Cycle-accurate** simulation and verification (*Verilator*)

Makefile based:

- Compilation (*Embench*) and software encryption flow
- **Cycle-accurate** simulation and verification (*Verilator*)
- FPGA design flow (*Vivado*)
- **Timing** simulation and verification (*Modelsim*)
- Execution and validation on FPGA board

Makefile based:

- Compilation (*Embench*) and software encryption flow
- **Cycle-accurate** simulation and verification (*Verilator*)
- FPGA design flow (*Vivado*)
- **Timing** simulation and verification (*Modelsim*)
- Execution and validation on FPGA board
- **FIA**: on FPGA (memory only) and on simulations

Validation and characterization

Cycle-accurate simulations

✓ Valid execution for 23/25 programs

At each cycle: PC and instruction in the fetch are compared with those of an unprotected simulation

PROGRAM_NAME	ENCRYPT	MODE	TEST	REASON END.	SIM_TIME	TIMESTAMP
crc32	instr	VERIF	SUCCESS	VALID EXEC	1149080	Fri Sep 20 11:32:33 2024
cubic	instr	VERIF	SUCCESS	VALID EXEC	1380282	Fri Sep 20 11:32:40 2024
dhystone	instr	VERIF	SUCCESS	VALID EXEC	611550	Fri Sep 20 11:32:43 2024
edn	instr	VERIF	SUCCESS	VALID EXEC	642402	Fri Sep 20 11:32:46 2024
fibonacci	instr	VERIF	SUCCESS	VALID EXEC	187694	Fri Sep 20 11:32:47 2024
huffbench	instr	VERIF	SUCCESS	VALID EXEC	904196	Fri Sep 20 11:32:51 2024
matmult-int	instr	VERIF	SUCCESS	VALID EXEC	839262	Fri Sep 20 11:32:54 2024
md5sum	instr	VERIF	SUCCESS	VALID EXEC	681136	Fri Sep 20 11:32:57 2024
minver	instr	VERIF	SUCCESS	VALID EXEC	785076	Fri Sep 20 11:33:00 2024
mont64	instr	VERIF	SUCCESS	VALID EXEC	515094	Fri Sep 20 11:33:03 2024
...						
tarfind	instr	VERIF	SUCCESS	VALID EXEC	717312	Fri Sep 20 11:33:51 2024
ud	instr	VERIF	SUCCESS	VALID EXEC	588562	Fri Sep 20 11:33:54 2024
wikisort	instr	VERIF	SUCCESS	VALID EXEC	3579694	Fri Sep 20 11:34:06 2024

Validation and characterization

Cycle-accurate simulations

✓ Valid execution for 23/25 programs

✗ Limitation of 12 successors of *jalr* for 2/25 programs

At each cycle: PC and instruction in the fetch are compared with those of an unprotected simulation

PROGRAM_NAME	ENCRYPT	MODE	TEST	REASON END.	SIM_TIME	TIMESTAMP
crc32	instr	VERIF	SUCCESS	VALID EXEC	1149080	Fri Sep 20 11:32:33 2024
cubic	instr	VERIF	SUCCESS	VALID EXEC	1380282	Fri Sep 20 11:32:40 2024
dhrystone	instr	VERIF	SUCCESS	VALID EXEC	611550	Fri Sep 20 11:32:43 2024
edn	instr	VERIF	SUCCESS	VALID EXEC	642402	Fri Sep 20 11:32:46 2024
fibonacci	instr	VERIF	SUCCESS	VALID EXEC	187694	Fri Sep 20 11:32:47 2024
huffbench	instr	VERIF	SUCCESS	VALID EXEC	904196	Fri Sep 20 11:32:51 2024
matmult-int	instr	VERIF	SUCCESS	VALID EXEC	839262	Fri Sep 20 11:32:54 2024
md5sum	instr	VERIF	SUCCESS	VALID EXEC	681136	Fri Sep 20 11:32:57 2024
minver	instr	VERIF	SUCCESS	VALID EXEC	785076	Fri Sep 20 11:33:00 2024
mont64	instr	VERIF	SUCCESS	VALID EXEC	515094	Fri Sep 20 11:33:03 2024
...						
tarfind	instr	VERIF	SUCCESS	VALID EXEC	717312	Fri Sep 20 11:33:51 2024
ud	instr	VERIF	SUCCESS	VALID EXEC	588562	Fri Sep 20 11:33:54 2024
wikisort	instr	VERIF	SUCCESS	VALID EXEC	3579694	Fri Sep 20 11:34:06 2024

Validation and characterization

Cycle-accurate simulations

✓ Valid execution for 23/25 programs

✗ Limitation of 12 successors of *jalr* for 2/25 programs ⇒ ✓ Solved with NOPs
(code and clock cycle overhead < 0.06%)

At each cycle: PC and instruction in the fetch are compared with those of an unprotected simulation

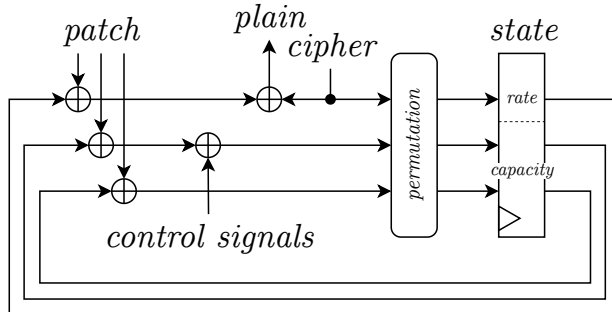
PROGRAM_NAME	ENCRYPT	MODE	TEST	REASON	END.	SIM_TIME	TIMESTAMP
crc32	instr	VERIF	SUCCESS	VALID	EXEC	1149080	Fri Sep 20 11:32:33 2024
cubic	instr	VERIF	SUCCESS	VALID	EXEC	1380282	Fri Sep 20 11:32:40 2024
dhrystone	instr	VERIF	SUCCESS	VALID	EXEC	611550	Fri Sep 20 11:32:43 2024
edn	instr	VERIF	SUCCESS	VALID	EXEC	642402	Fri Sep 20 11:32:46 2024
fibonacci	instr	VERIF	SUCCESS	VALID	EXEC	187694	Fri Sep 20 11:32:47 2024
huffbench	instr	VERIF	SUCCESS	VALID	EXEC	904196	Fri Sep 20 11:32:51 2024
matmult-int	instr	VERIF	SUCCESS	VALID	EXEC	839262	Fri Sep 20 11:32:54 2024
md5sum	instr	VERIF	SUCCESS	VALID	EXEC	681136	Fri Sep 20 11:32:57 2024
minver	instr	VERIF	SUCCESS	VALID	EXEC	785076	Fri Sep 20 11:33:00 2024
mont64	instr	VERIF	SUCCESS	VALID	EXEC	515094	Fri Sep 20 11:33:03 2024
...							
tarfind	instr	VERIF	SUCCESS	VALID	EXEC	717312	Fri Sep 20 11:33:51 2024
ud	instr	VERIF	SUCCESS	VALID	EXEC	588562	Fri Sep 20 11:33:54 2024
wikisort	instr	VERIF	SUCCESS	VALID	EXEC	3579694	Fri Sep 20 11:34:06 2024

Validation and characterization FPGA

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

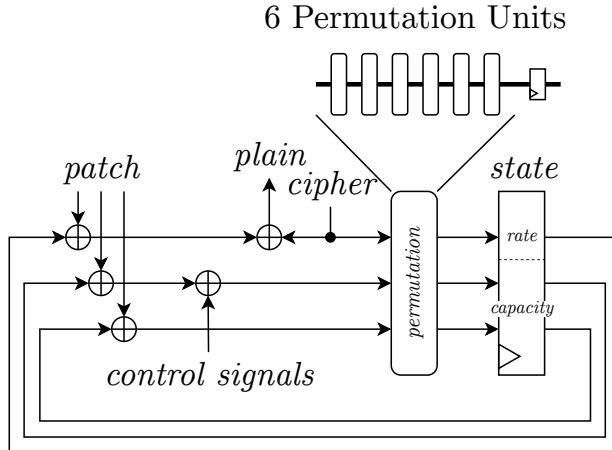
Validation and characterization

FPGA: Cross domain clocking



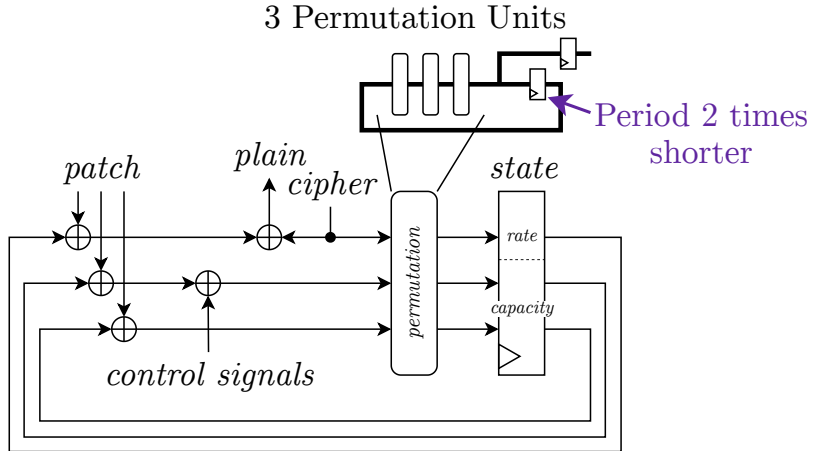
Validation and characterization

FPGA: Cross domain clocking



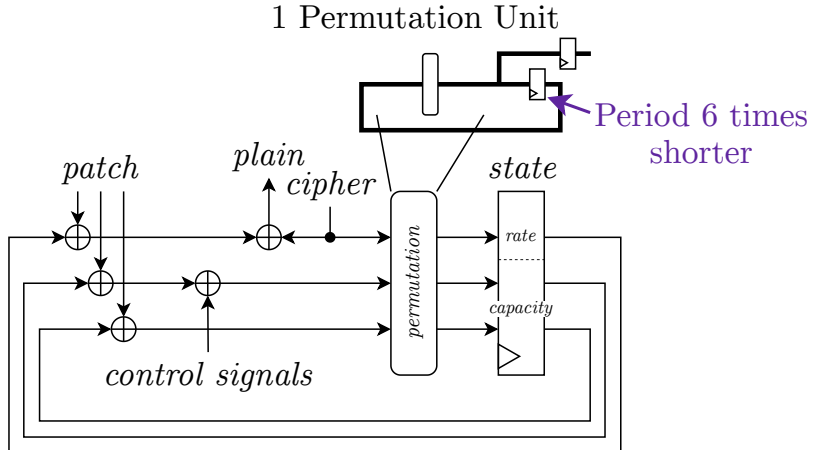
Validation and characterization

FPGA: Cross domain clocking

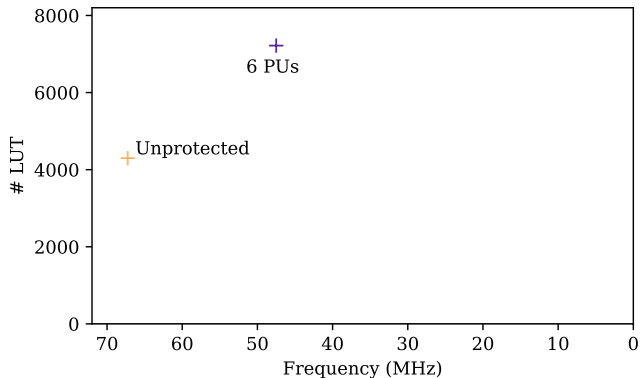


Validation and characterization

FPGA: Cross domain clocking

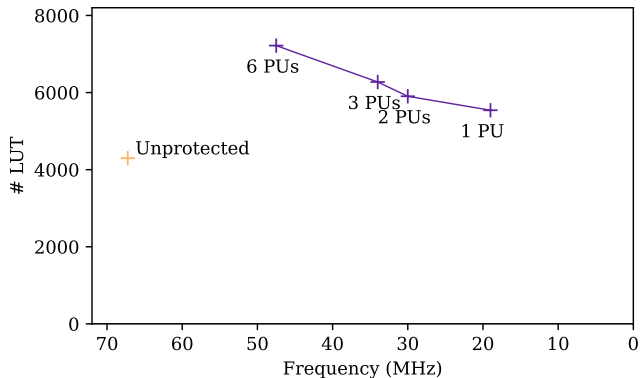


Validation and characterization FPGA



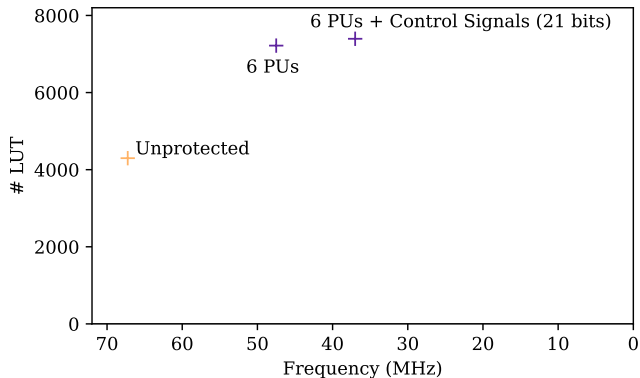
Versions	Max. Freq. (MHz)	# LUT	# FF
Unprotected	67.45	4298	2064
6 PUs	47.5 (-29.6%)	7218 (+67.9%)	3402 (+64.8%)

Validation and characterization FPGA



Versions	Max. Freq. (MHz)	# LUT	# FF
Unprotected	67.45	4298	2064
6 PUs	47.5 (-29.6%)	7218 (+67.9%)	3402 (+64.8%)
3 PUs	34.0 (-49.6%)	6274 (+46.0%)	3733 (+80.9%)
2 PUs	30.0 (-55.5%)	5904 (+37.4%)	3733 (+80.9%)
1 PU	19.0 (-71.8%)	5541 (+28.9%)	3733 (+80.9%)

Validation and characterization FPGA



Versions	Max. Freq. (MHz)	# LUT	# FF
6 PUs	47.5	7218	3402
6 PUs + Control Signals (21 bits)	37 (-22%)	7395 (+2.5%)	3435 (+0.9%)

Name	Memory	Cycles
(Ascon 320bits)	1047%	0%

Configuration of control signals:

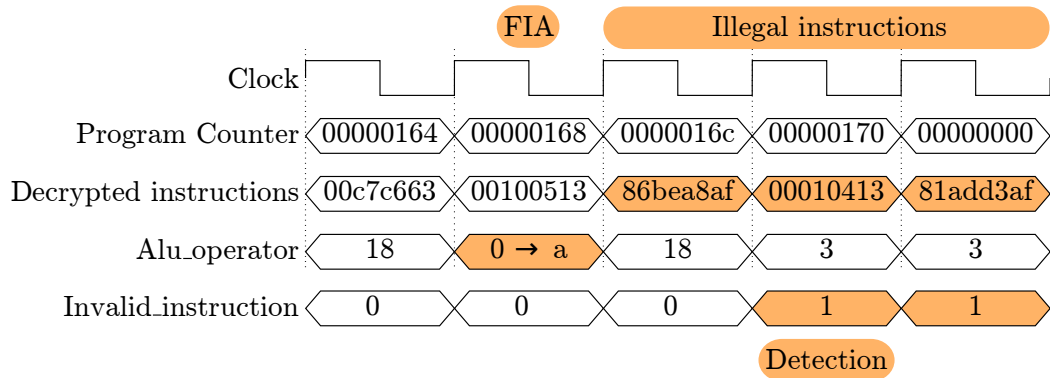
- 5 bits from execute
- 5 bits from write-back

Validation and characterization FPGA

Name	Memory (%)	Cycles (%)	Frequency (%)	Utilization (%)
INSTR. ENC.	2–38	129–293*	0	26 ALM
CONFIDAENT	116–218*	151–276*		
SOFIA	110–437	36–438	–23	12 LUT
CCFI-CACHE	118–160	2–63		11 LUT 9 FF
CIFER	16–68	0	0	35–55 Slice
SOFT-ONLY	9–3550*	92–3100*	0	<i>null</i>
SECDEC	1–2	1–2		4–17 GE*
HAPEI	123–507*			
SCFP	15–26	4–15		47 GE
GPSA-CSM	88–118*	2–71*		6 GE
MAFIA	7–55	3–44		7–24 GE
This work				
(Ascon 320bits)	1047	0	–30	29 LUT 81 FF
(64bits)	247	0		

Validation and characterization

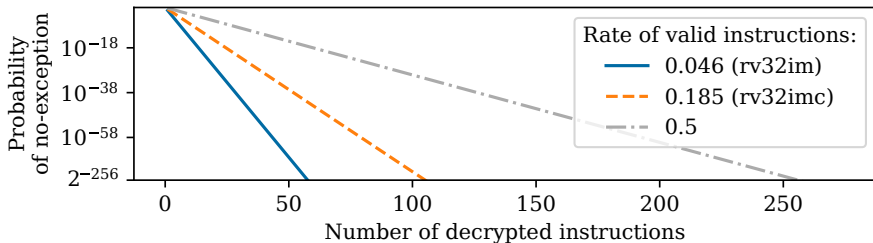
Experimental validation



Validation and characterization

Probability of no-detection

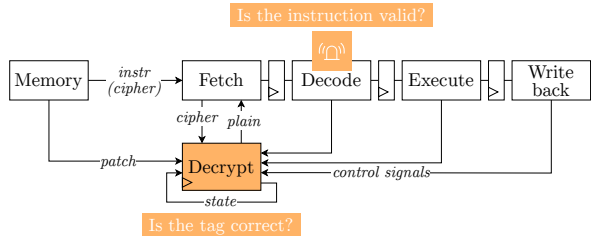
How many random instructions before detection?



- 1 Who am I?
- 2 Context for execution integrity
- 3 Proposed scheme
- 4 Validation and characterization
- 5 Conclusion**

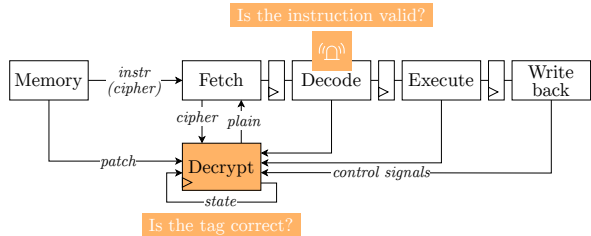
Chained Instruction Encryption with Associated Control Signals:

- ✓ **Confidentiality** of Instructions
- ✓ **Integrity** of Control Flow
- ✓ **Integrity** of Instructions
- ✓ **Integrity** of Deterministic Control Signals



Chained Instruction Encryption with Associated Control Signals:

- ✓ **Confidentiality** of Instructions
 - ✓ **Integrity** of Control Flow
 - ✓ **Integrity** of Instructions
 - ✓ **Integrity** of Deterministic Control Signals
-
- ✓ Zero clock cycle penalty

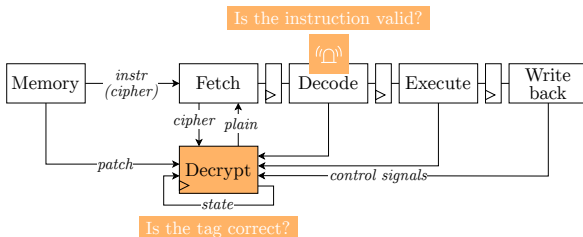


Chained Instruction Encryption with Associated Control Signals:

- ✓ **Confidentiality** of Instructions
- ✓ **Integrity** of Control Flow
- ✓ **Integrity** of Instructions
- ✓ **Integrity** of Deterministic Control Signals

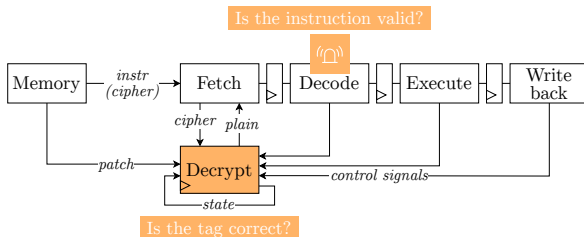
- ✓ Zero clock cycle penalty

- No cache
- Fetch instruction 1 cycle



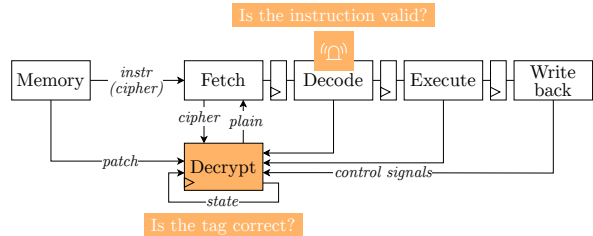
Chained Instruction Encryption with Associated Control Signals:

- ✓ Conception
- ✓ Implementation → cv32e40p
- ✓ Validation and Characterization
- ✓ **Submission:** HOST 2025
- ✓ **Open-source:** Github



Chained Instruction Encryption with Associated Control Signals:

- ✓ Conception
- ✓ Implementation → cv32e40p
- ✓ Validation and Characterization
- ✓ **Submission:** HOST 2025
- ✓ **Open-source:** Github



In progress:

- Reduce patch size: Memory overhead ↘
- Hardware demo
- ➔ Manuscript

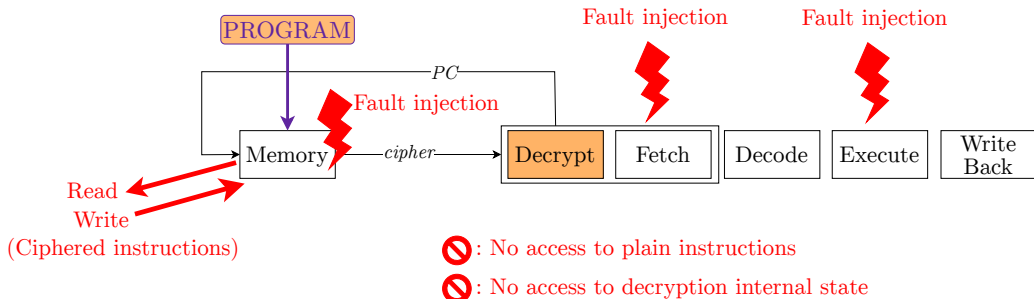


- [1] François Bonnal, Vincent Dupaquis, Olivier Potin, and Jean-Max Dutertre.
Software-only control-flow integrity against fault injection attacks.
In 2023 26th Euromicro Conference on Digital System Design (DSD), pages 269–277. IEEE, 2023.
- [2] Thomas Chamelot, Damien Couroussé, and Karine Heydemann.
Mafia: Protecting the microarchitecture of embedded systems against fault injection attacks.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023.
- [3] Jean-Luc Danger, Adrien Facon, Sylvain Guilley, Karine Heydemann, Ulrich Kühne, Abdelmalek Si Merabet, and Michaël Timbert.
Ccfi-cache: A transparent and flexible hardware protection for code and control-flow integrity.
In 2018 21st Euromicro Conference on Digital System Design (DSD), pages 529–536. IEEE, 2018.
- [4] Ruan De Clercq, Johannes Götzfried, David Übler, Pieter Maene, and Ingrid Verbauwhede.
Sofia: Software and control flow integrity architecture.
Computers & Security, 68:16–35, 2017.
- [5] Thomas Hiscock, Olivier Savry, and Louis Goubin.
Lightweight instruction-level encryption for embedded processors using stream ciphers.
Microprocessors and Microsystems, 64:43–52, 2019.
- [6] Ronan Lashermes, Hélène Le Boudier, and Gaël Thomas.
Hardware-assisted program execution integrity: Hapei.
In Secure IT Systems: 23rd Nordic Conference, NordSec 2018, Oslo, Norway, November 28-30, 2018, Proceedings 23, pages 405–420. Springer, 2018.
- [7] Johan Laurent, Vincent Berouille, Christophe Deleuze, Florian Pebay-Peyroula, and Athanasios Papadimitriou.
Cross-layer analysis of software fault models and countermeasures against hardware fault attacks in a risc-v processor.
Microprocessors and Microsystems, 71:102862, 2019.

- [8] Gaëtan Leplus, Olivier Savry, and Lilian Bossuet.
Secdec: Secure decode stage thanks to masking of instructions with the generated signals.
In *2022 25th Euromicro Conference on Digital System Design (DSD)*, pages 556–563. IEEE, 2022.
- [9] Olivier Savry, Mustapha El-Majihi, and Thomas Hiscock.
Confidaent: Control flow protection with instruction and data authenticated encryption.
In *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pages 246–253. IEEE, 2020.
- [10] Mario Werner, Thomas Unterluggauer, David Schaffenrath, and Stefan Mangard.
Sponge-based control-flow protection for iot devices.
In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 214–226. IEEE, 2018.
- [11] Mario Werner, Erich Wenger, and Stefan Mangard.
Protecting the control flow of embedded processors against fault attacks.
In *Smart Card Research and Advanced Applications: 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers 14*, pages 161–176. Springer, 2016.
- [12] Bilgiday Yuce, Patrick Schaumont, and Marc Witteman.
Fault attacks on secure embedded software: Threats, design, and evaluation.
Journal of Hardware and Systems Security, 2:111–130, 2018.
- [13] Anthony Zgheib, Olivier Potin, Jean-Baptiste Rigaud, and Jean-Max Dutertre.
Cifer: Code integrity and control flow verification for programs executed on a risc-v core.
In *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 100–110. IEEE, 2023.

Extra slides

Extra slides



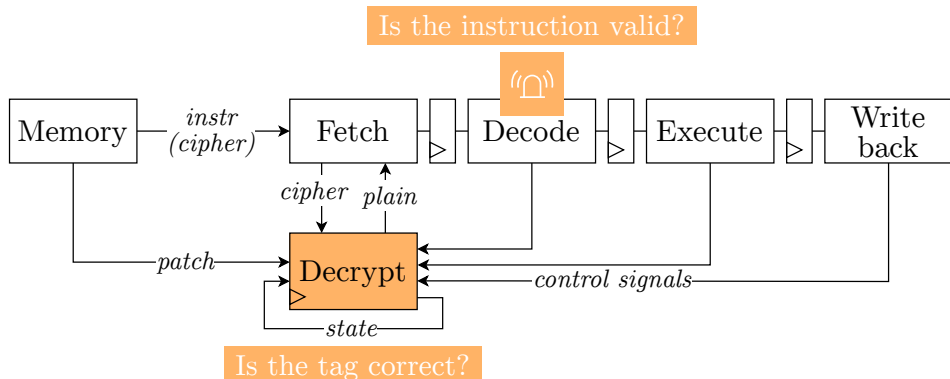
Extra slides

RISC-V core: cv32e40p

OpenHW group:

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

Chained Instruction Encryption with Associated Control Signals



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

Extra slides

Authenticated Encryption: ASCON

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

Extra slides

Authenticated Encryption: ASCON

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

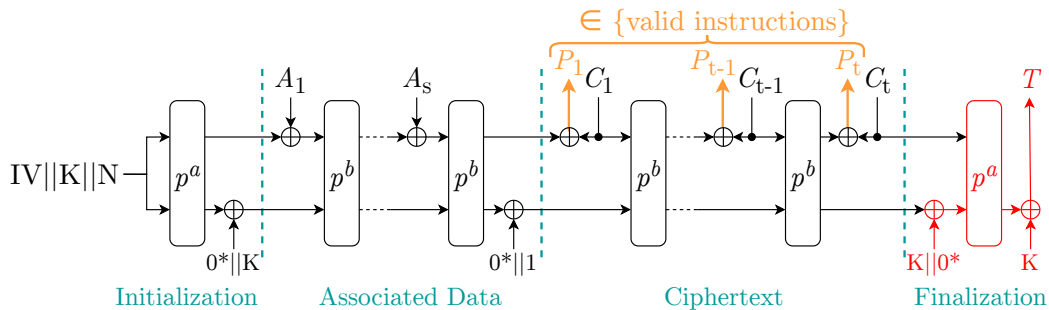
- Winner of CAESAR Authenticated Encryption - Lightweight Cryptography (2019)
- Winner of NIST - Lightweight Cryptography (2023) ⇒ **standardize the ASCON family**
- Lightweight (better Throughput per Area than AES [1])
- Highly tested
- Large security margins

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

Extra slides

Authenticated Encryption: ASCON

ASCON: Use-case of restricted set of valid data

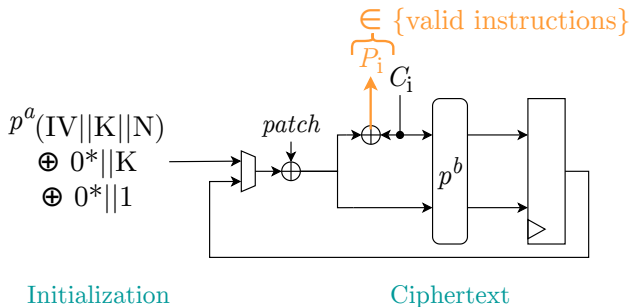


ASCON Decryption scheme

Extra slides

Authenticated Encryption: ASCON

ASCON: Authenticated Encryption provided from restricted set of valid instructions

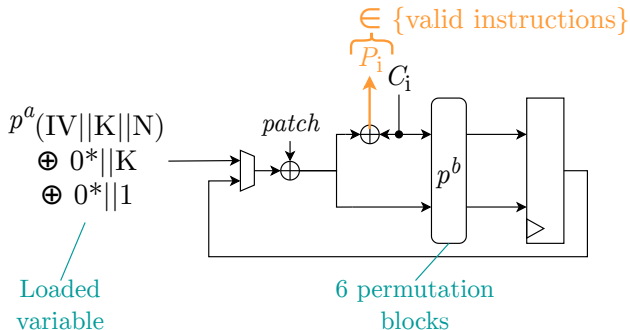


ASCON Decryption scheme

Extra slides

Authenticated Encryption: ASCON

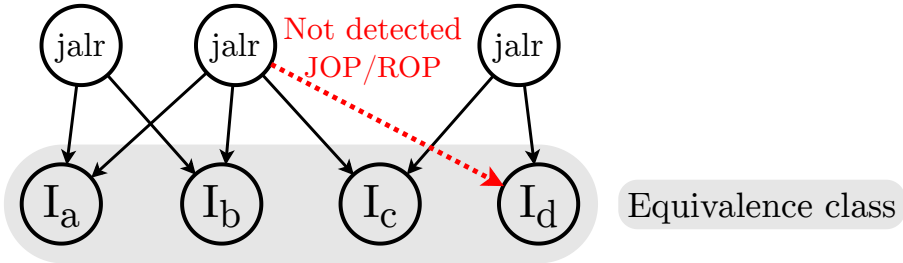
ASCON: Authenticated Encryption provided from restricted set of valid instructions

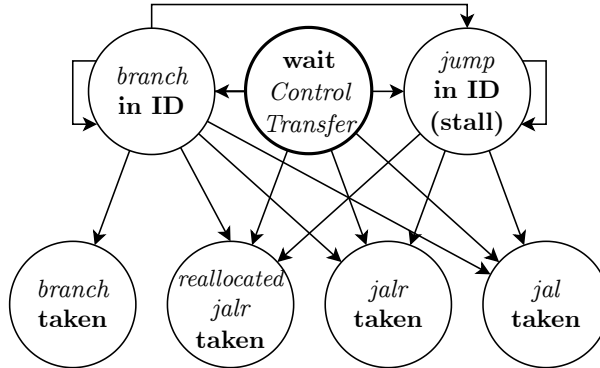


ASCON Decryption scheme

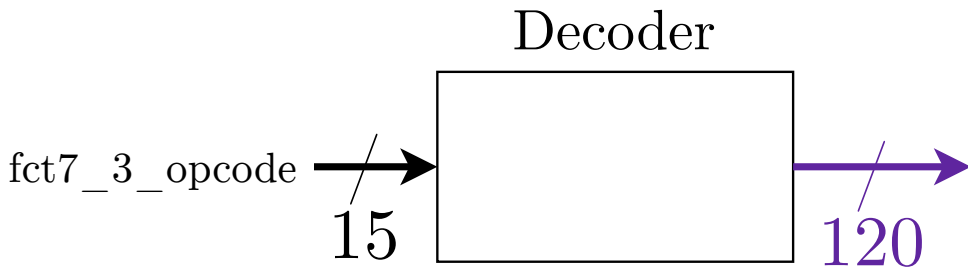
Extra slides

Equivalence class





Every state can reach "*wait Control Transfer*" except "*jump in ID (stall)*".



fct7_3_opcode

control_signals

$\left(\begin{array}{l} 0, \text{e1060000080116,80080010da0,40} \\ 1, \text{e000c000080100,801800109a0,0} \\ 2, \text{e000c000080100,801800109a0,0} \\ 3, \text{e000c000080100,40800800109a0,0} \\ 4, \text{e1060000080100,80080010db8,0} \end{array} \right)$

TABLE A.1 – Primitives cryptographiques offrant un service donné

Service		Cryptographie symétrique	Cryptographie asymétrique
Confidentialité		Chiffrement conventionnel par bloc (A.1.1.1)	Chiffrement à clé publique (A.2.1)
		ou par flot (A.1.1.2)	Échange de clé (A.2.3)
Intégrité		Code d'authentification de message (A.1.3)	Signature numérique (A.2.2)
Authentification	de données		
	d'entités	Défi-réponse (A.1.4)	
Non-répudiation		Aucune primitive	



MINES
Saint-Étienne

Une école de l'IMT

Code Encryption for Confidentiality and Execution Integrity down to Control Signals

Théophile Gousselot

Ph.D. Student

SemSecuElec - Rennes, Inria

Oct 18, 2024

