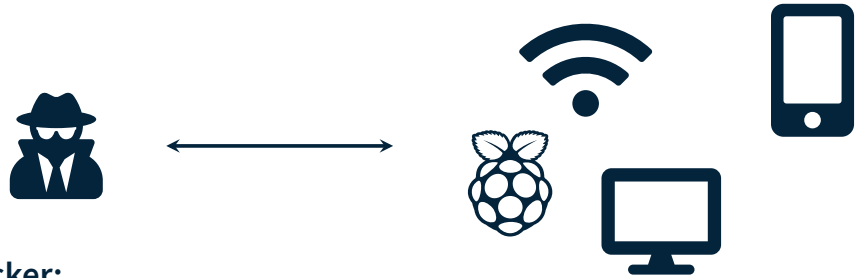
Hardware Extensions and Compiler Support for Protection Against Fault Attacks

Robert Schilling

September 7, 2023

We live in a connected world

Attack Setup



- **Attacker:**
  - Has **physical** access to device
  - Has **remote** access to device

Contribution

Data Protection:

- Encryption with fault-protection on algorithmic level
- Energy-efficient encryption for IoT

Control-Flow:

- Software-based CFI against faults
- Hardening the syscall interface
- Protected conditional branches

Memory Accesses:

- Pointer encoding and linked memory access
- Secure page table walk

Data Protection

Motivation

- **Encryption** is a major building block for secure systems
- Exposed devices require dedicated countermeasures to protect against physical attacks
  - Requires additional logic $\rightarrow$ **reduces** energy-efficiency
  - Important for energy-constraint IoT end-nodes
- Can we provide energy-efficient encryption that is fault-protected by design?

An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics

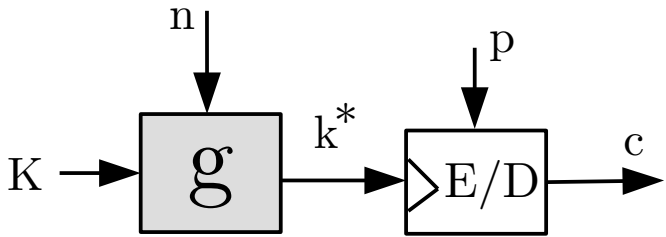
Francesco Conti, **Robert Schilling**, Pasquale Davide Schiavone, Antonio Pullini, Davide Rossi, Frank Kagan Gürkaynak, Michael Muehlberghuber, Michael Gautschi, Igor Loi, Germain Haugou, Stefan Mangard, and Luca Benini. **An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics.** *IEEE Trans. Circuits Syst. I Regul. Pap.* 64-I (2017), pp. 2481–2494

High Speed ASIC Implementations of Leakage-Resilient Cryptography

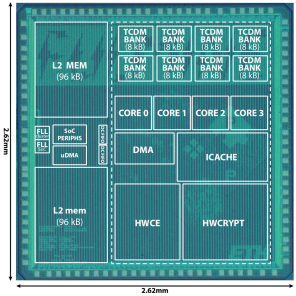
Robert Schilling, Thomas Unterluggauer, Stefan Mangard, Frank K. Gürkaynak, Michael Muehlberghuber, and Luca Benini. **High speed ASIC implementations of leakage-resilient cryptography.** 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018. IEEE, 2018, pp. 1259–1264

Fresh Re-keying to counteract Fault Attacks

- Break the assumptions of DFA attacks $\rightarrow$ use a new key for every encryption
- Use a re-keying function that always computes a fresh session key based on a nonce



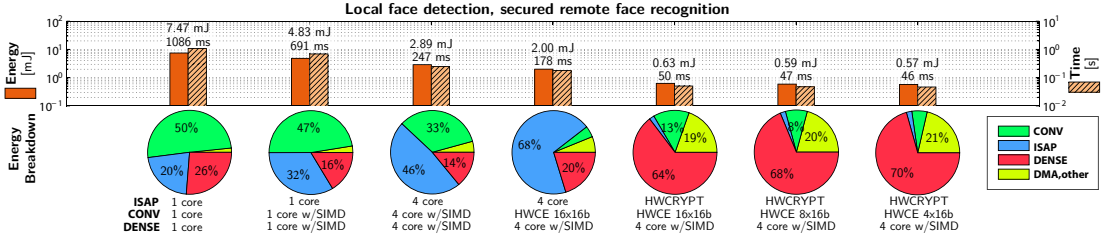
Fulmine and HWCRYPT



Fulmine SoC in UMC 65nm LL 1P8M

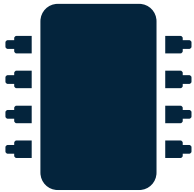
- 4-core RISC SoC with two hardware accelerators
- **HWCE** Convolutional accelerator engine
- **HWRYPT** Cryptographic accelerator engine
  - Polymul + AES-2PRG (100 pJ/b, 67 Gbit/s/W)
  - ISAP (70 pJ/b, 100 Gbit/s/W)

Secure Face Detection



- Local face detection with offline face recognition
- After detection $\rightarrow$ image is encrypted and sent to the host
- 24$\times$ speedup and 13$\times$ reduction in energy
- 5.74 pJ/op efficiency with on equivalent RISC operations

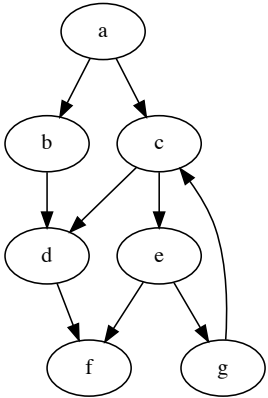
Summary



- Energy-efficient encryption possible on the architectural level
  - Without aggressive voltage and technology scaling
  - No dedicated countermeasures needed due to algorithmic protection
- End-to-end applications show the efficiency of tightly-coupled hardware accelerators

Control-Flow Integrity and Friends

Motivation

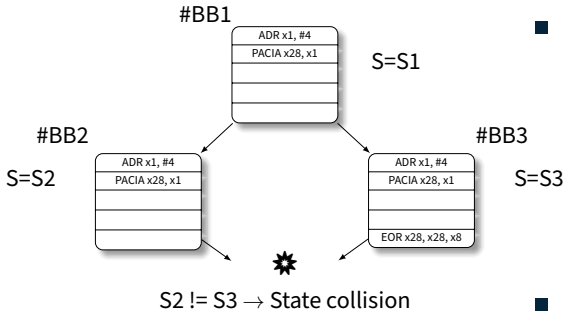


- Why do we need another CFI countermeasure?
  - **Combined** software- and fault attacks bypass existing schemes
  - Strong CFI requires hardware support $\rightarrow$ **not** possible for commodity systems

FIPAC: Thwarting Fault- and Software-Induced Control-Flow Attacks with ARM Pointer Authentication

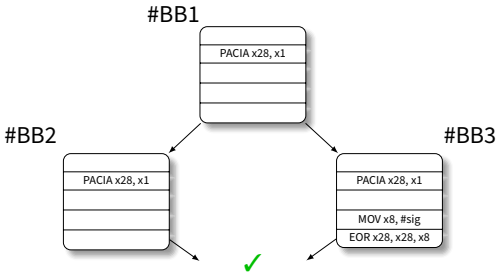
Robert Schilling, Pascal Nasahl, and Stefan Mangard. **FIPAC: Thwarting Fault- and Software-Induced Control-Flow Attacks with ARM Pointer Authentication.** Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings. Springer, 2022, pp. 100–124

Design of FIPAC



- Cryptographically enforced CFI with ARM Pointer Authentication
  - Every basic block updates a global CFI state with PAC
  - Unique CFI state per basic block
- Enforcement of the control-flow graph at basic-block level
- Checks at different locations

Design of FIPAC

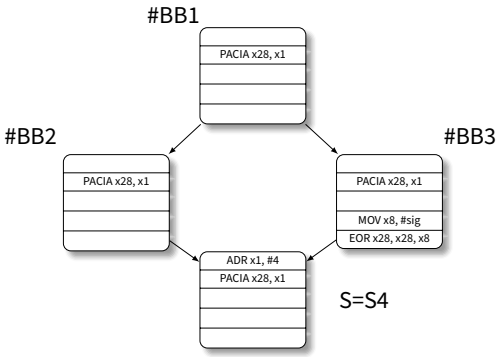


Insert a justifying signature

Sig = S2 ⊕ S3

- Cryptographically enforced CFI with ARM Pointer Authentication
  - Every basic block updates a global CFI state with PAC
  - Unique CFI state per basic block
- Enforcement of the control-flow graph at basic-block level
- Checks at different locations

Design of FIPAC



- Cryptographically enforced CFI with ARM Pointer Authentication
  - Every basic block updates a global CFI state with PAC
  - Unique CFI state per basic block
- Enforcement of the control-flow graph at basic-block level
- Checks at different locations

Checking Strategy

- Where to place a check?
  - More checks $\rightarrow$ better detection latency ✔
  - More checks $\rightarrow$ more overhead ✘
- 3 strategies implemented
  1. Single check at program end
  2. Check at every basic block
  3. Check at function end
- Custom strategies possible

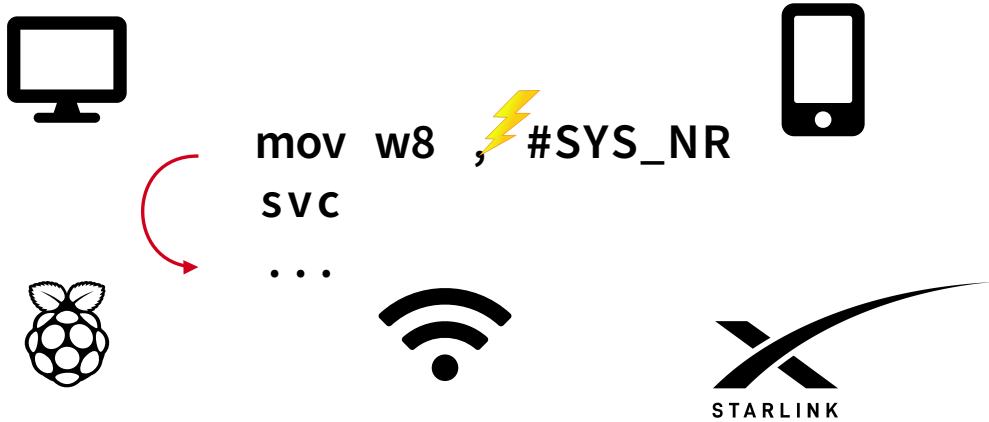
Implementation and Summary

- **Open source** LLVM toolchain[1] to automatically instrument arbitrary C-code with state updates and checks
- Runtime evaluation on RaspberryPi with PAC emulation
- **Runtime overhead on SPEC2017**
  - 19 %: Single check at program end
  - 63 %: Check on every basic block
  - 22 %: Check on function end

[1] https://github.com/Fipac/Fipac

Can we extend the scope of protection of FIPAC?

Syscalls are Everywhere



SFP: Providing System Call Flow Protection against Software and Fault Attacks

Robert Schilling, Pascal Nasahl, Martin Unterguggenberger, and Stefan Mangard. **SFP: Providing System Call Flow Protection against Software and Fault Attacks.** *CoRR* abs/2301.02915 (2023)

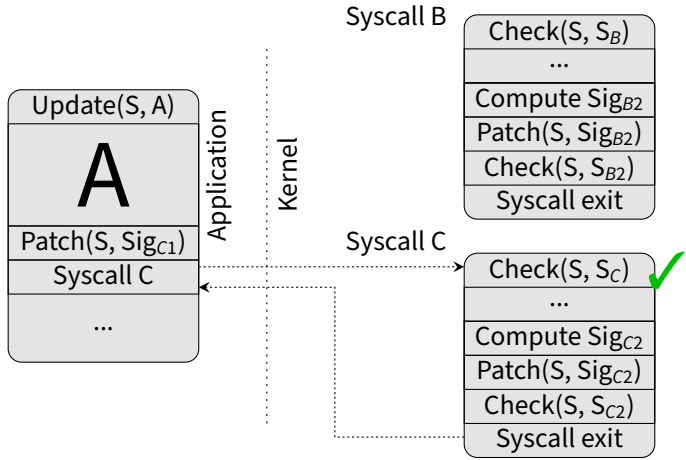
Requirements for System Call Protection

1. Prevent an attacker from **manipulating** the system call number
2. Ensure that a system call cannot be **skipped**
3. Ensure the system call dispatcher in the kernel executes the **correct** syscall function

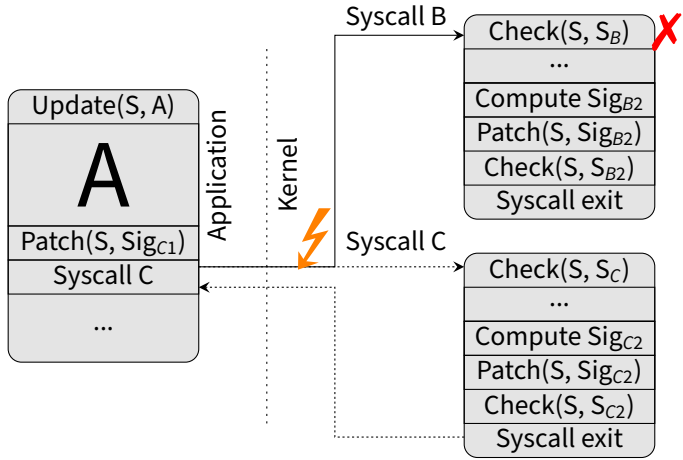
Design Idea of SFP

- **Link** the system call to the cryptographic CFI state
  - Bind the syscall to the CFI state ahead of its execution
- Dynamically verify the correct syscall function in the kernel
- Perform CFI checks when **entering** and **leaving** the kernel

SFP Design Summary



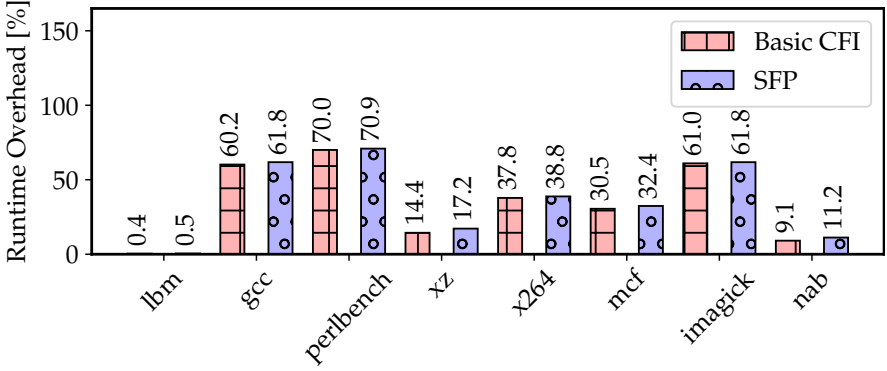
SFP Design Summary



Implementation

- Modified C standard library
  - Patch all system call invocations with patch sequence
  - Manual process with source code modifications
- Modified Linux kernel
  - CFI checking policy
  - System call protection (2nd stage linking)
  - Dynamic CFI instrumentation

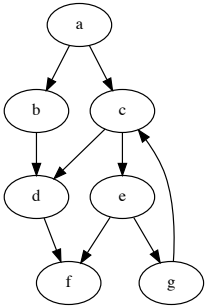
Runtime Overhead for SPEC2017



- SFP increases the runtime overhead by **1.8 %** on top of FIPAC
- From **18.8 %** (FIPAC) to **20.6 %** (with SFP)

What about branches?

Limitations of Existing CFI Protection Schemes



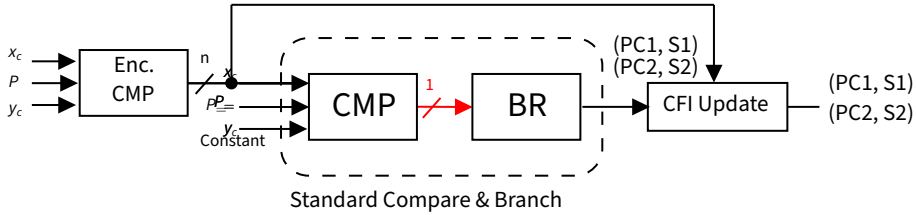
- CFI protects the **control-flow graph (CFG)**
- All branch successors are within the CFG
- Point when data touches control-flow control remains **unprotected**
- Conditional branches are **unprotected**
  - Critical decisions depend on conditional branches

Securing Conditional Branches in the Presence of Fault Attacks

Robert Schilling, Mario Werner, and Stefan Mangard. **Securing conditional branches in the presence of fault attacks.** 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018. IEEE, 2018, pp. 1586–1591

Generic Protected Conditional Branches

- Multiple attack vectors to bypass conditional branches
  1. Faulting the operands → Add redundancy to x and y (AN-codes)
  2. Faulting the comparison → Encoded comparison in software
  3. Faulting the branch → Link the redundant condition value with the CFI state

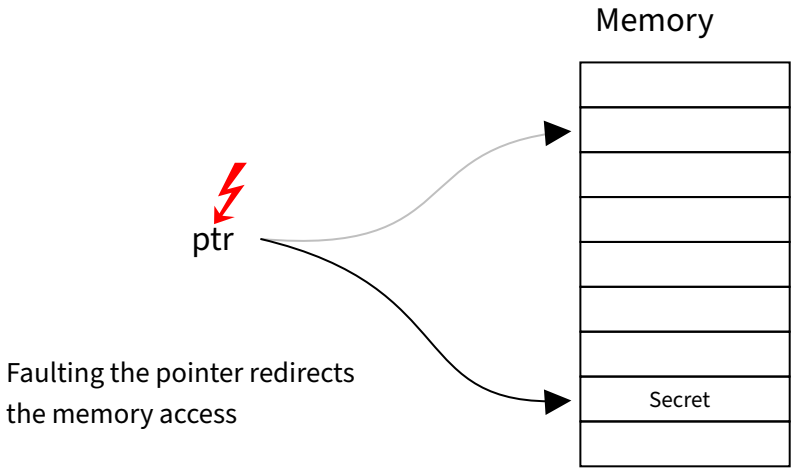


Integration and Tooling

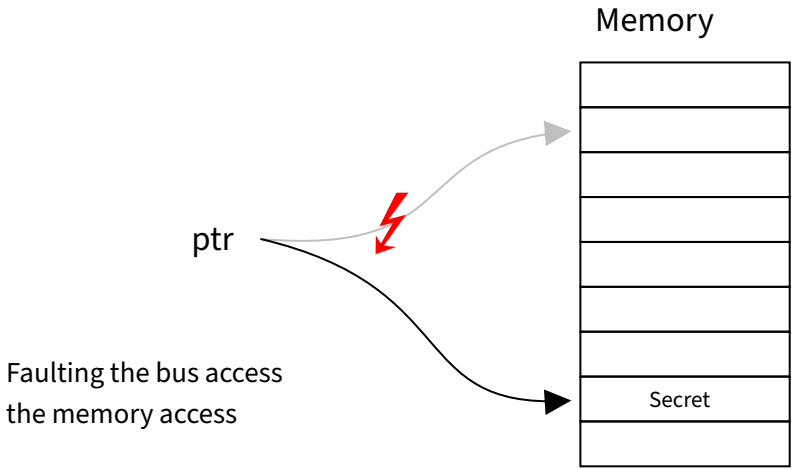
- Protected comparison operations with AN-codes for all predicates
- LLVM compiler integration in the middle-end
  - Target independent encoded comparison instrumentation
  - Only CFI state update is target dependent
- Hardware prototype using on ARM-based software-centric CFI scheme
- Prototype secure boot application with negligible runtime overhead
- Generic approach compatible with FIPAC

Memory Access Protection

Fault Attacks on Memory Accesses



Fault Attacks on Memory Accesses

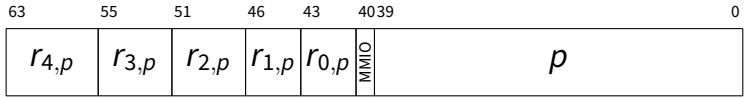


Pointing in the Right Direction - Securing Memory Accesses in a Faulty World

Robert Schilling, Mario Werner, Pascal Nasahl, and Stefan Mangard. **Pointing in the Right Direction - Securing Memory Accesses in a Faulty World.** Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018. ACM, 2018, pp. 595–604

Multi-Residue Encoded Pointers

- Reduce address space to 40-bits
- Upper 24-bits store redundancy information and MMIO indicator bit
  - Multi-residue code with 5-bit Hamming distance



- RISC-V instruction set extension for encoded pointer arithmetic

Secure Memory Access

- Pointers are protected but memory access still can be redirected
- Establish a link between the redundant address and redundant data
- Perform a linking overlay on top of encoded data
- Unlinking operation only successful when using the correct pointer and correct memory access
  - Translate **addressing** errors to **data** errors
- Byte-granular XOR with compressed pointer $\rightarrow$ Support for byte-granular data access

Implementation and Results

Hardware implementation on 32-bit CV32E40P RISC-V Processor

- Hardware extension with instructions for encoded pointer arithmetic and linked memory accesses

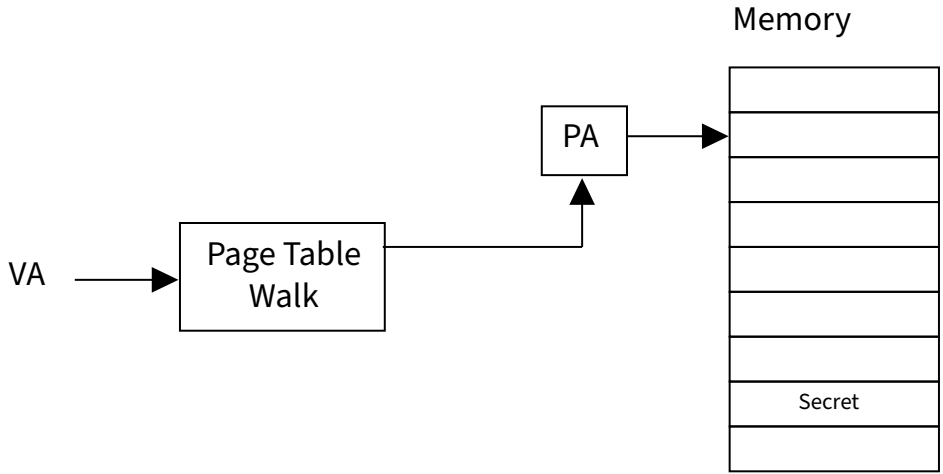
LLVM compiler integration

- Automatically protects all pointer arithmetic and memory accesses

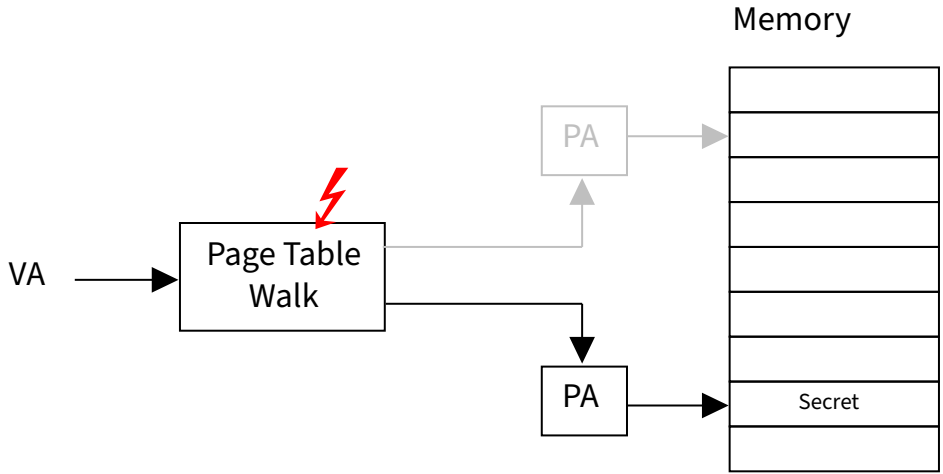
- **Code overhead:**
  - 9.9 % on average
- **Runtime overhead:**
  - 6.3 % on average

What about larger systems?

Fault Attacks on the Page Table Walk



Fault Attacks on the Page Table Walk

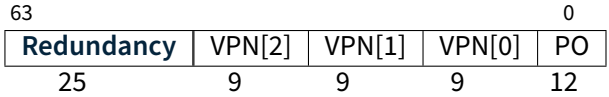


SecWalk: Protecting Page Table Walks Against Fault Attacks

Robert Schilling, Pascal Nasahl, Stefan Weiglhofer, and Stefan Mangard. **SecWalk: Protecting Page Table Walks Against Fault Attacks.** IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2021, Tysons Corner, VA, USA, December 12-15, 2021. IEEE, 2021, pp. 56–67

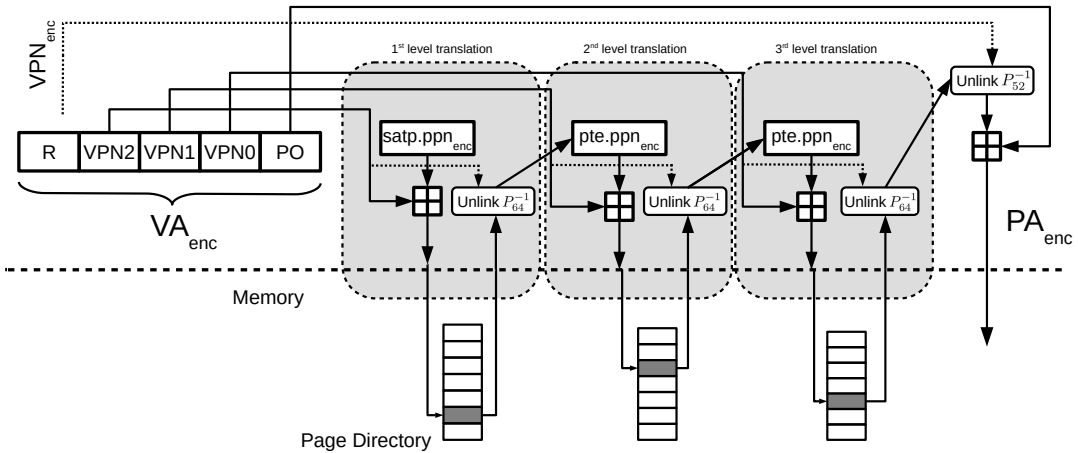
Secure Page Table Walk

- **Translates** virtual address to physical → Used for the memory access
- Virtual address grouped to 27-bit VPN and 12-bit page offset



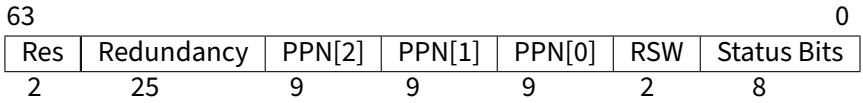
- Page table walk translates 27-bit VPN to a 44-bit PPN
- Page offset remains untranslated and added to the PPN
- **Idea:** Faulty translations detectable through redundancy properties

Protected Page Table Walk

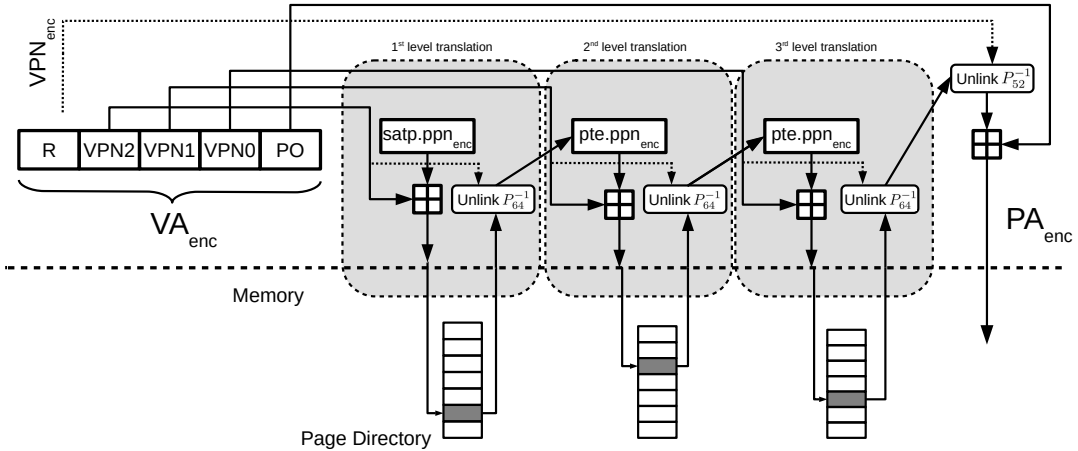


Protected Page Table Entries

- **Encode** PPN with a multi-residue code
- Unused PTE bits store **redundancy** information
- Still access to parts of the PPN for the page table walk



Protected Page Table Walk



Implementation and Results

- **Hardware implementation** on 64-bit CVA6 RISC-V Processor
  - Hardware page table walker and custom instructions for linking
- **Compiler integration**
  - Re-use the compiler from before
- **Operating system integration to seL4**
  - Creates linked page-directory on start using custom instructions
  - **Microbenchmark:**
    - 11.05 % code overhead
    - 7.17 % runtime overhead
  - **OS integration to seL4:**
    - 13.1 % code overhead
    - 11.6 % runtime overhead

Conclusion

Summary

- Protection of different subsystems against fault attacks
  - Data protection: **Energy-efficient encryption with Fulmine**
  - Control-flow: **FIPAC, SFP, Conditional branches**
  - Memory accesses: **Pointer encoding and linked memory accesses, SecWalk**
- Transparent hardware changes do not influence the software
- Integrated toolchains support adoption on larger scale
- Research prototypes show reasonable overheads

Outlook

- Integration to larger systems
  - Selectively protect parts of the system, *i.e.*, a TEE
  - Useful for confidential computing platforms
- Use formal methods with fault injection
  - Generate proofs to state the level of security
- Data protection and computation
- Attestation and Licensing

The PhD in Numbers

- 18 (co-)authored papers in total
- 7 papers in this thesis
- Contributed to 2 lectures
  - Digital System Design (since WT2017)
  - Computer Organization (ST2019) $\rightarrow$ Computer Organization and Networks (since WT2019)
- Supervised 13 student projects
  - 2 internships, 3 bachelor theses
  - 3 master projects
  - 5 master theses

Hardware Extensions and Compiler Support for Protection Against Fault Attacks

Robert Schilling

September 7, 2023

Bibliography I

[CSS+17] Francesco Conti, **Robert Schilling**, Pasquale Davide Schiavone, Antonio Pullini, Davide Rossi, Frank Kagan Gürkaynak, Michael Muehlberghuber, Michael Gautschi, Igor Loi, Germain Haugou, Stefan Mangard, and Luca Benini. **An IoT Endpoint System-on-Chip for Secure and Energy-Efficient Near-Sensor Analytics.** *IEEE Trans. Circuits Syst. I Regul. Pap.* 64-I (2017), pp. 2481–2494.

[SNM22] **Robert Schilling**, Pascal Nasahl, and Stefan Mangard. **FIPAC: Thwarting Fault- and Software-Induced Control-Flow Attacks with ARM Pointer Authentication.** Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings. Springer, 2022, pp. 100–124.

Bibliography II

[SNUM23] **Robert Schilling**, Pascal Nasahl, Martin Unterguggenberger, and Stefan Mangard. **SFP: Providing System Call Flow Protection against Software and Fault Attacks.** *CoRR* abs/2301.02915 (2023).

[SNWM21] **Robert Schilling**, Pascal Nasahl, Stefan Weiglhofer, and Stefan Mangard. **SecWalk: Protecting Page Table Walks Against Fault Attacks.** IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2021, Tysons Corner, VA, USA, December 12-15, 2021. IEEE, 2021, pp. 56–67.

[SUM+18] **Robert Schilling**, Thomas Unterluggauer, Stefan Mangard, Frank K. Gürkaynak, Michael Muehlberghuber, and Luca Benini. **High speed ASIC implementations of leakage-resilient cryptography.** 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018. IEEE, 2018, pp. 1259–1264.

Bibliography III

[SWM18] **Robert Schilling**, Mario Werner, and Stefan Mangard. **Securing conditional branches in the presence of fault attacks.** 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018. IEEE, 2018, pp. 1586–1591.

[SWNM18] **Robert Schilling**, Mario Werner, Pascal Nasahl, and Stefan Mangard. **Pointing in the Right Direction - Securing Memory Accesses in a Faulty World.** Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018. ACM, 2018, pp. 595–604.