Tampere University

System-on-Chip Hardware Security

COMP.CE.250 System-on-Chip Design

Tom Szymkowiak

Outline

- Motivation
- Basic Concepts
- SoC Architecture Recap
- Threat-Based Design
- Attack Surfaces
- Mitigations
- Example Projects
- Research Areas
- Questions

Disclaimer – This is a very broad topic…

We can only cover so much

Motivation

Why is hardware security important?

Motivation – Security Definition

- Cyber Security Definition:
  - "*The approach and actions associated with security risk management processes followed by organizations and states to protect confidentiality, integrity and availability of data and assets used in cyber space. The concept includes guidelines, policies and collections of safeguards, technologies, tools and training to provide the best protection for the state of the cyber environment and its users.*" [SC17]
- Security should be treated as a primary concern for any engineering project i.e. with a similar rigor to safety.
- Safety vs Security:
  - Safety provides protection to people.
  - Security provides protection from people.

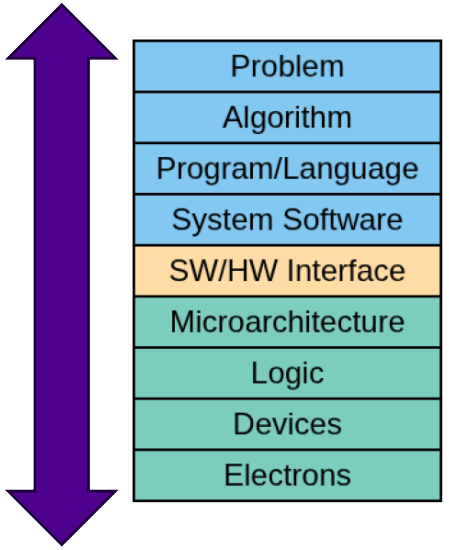
https://link.springer.com/article/10.1007/s38311-017-0152-7

Motivation – Why not only SW security?

- Increased need for HW-based security features due to:
  - Increasing size and complexity of software running on SoCs leads to increased probability of bugs and exploits within SW (Linux Kernel was 35.1 million lines of code in 2022 and an average of 223 vulnerabilities were identified each year 2012-2022 [JI24]).
  - Increasing availability of knowledge and test equipment which can be used to exploit side-channel vulnerabilities in HW is increasing.
  - There are physical attacks which cannot be protected by software alone.
- Software cannot protect itself from an attacker who controls the hardware beneath it.
- Hardware security protections are more difficult to change than SW.
- Energy efficiency and performance of system is improved by utilizing hardware-based security functions vs SW [SZ20].

Motivation – Why research this?

- “Attacks always get better, they never get worse.” [Schneier]
- Attack surface is rapidly expanding with proliferation of embedded devices using SoCs (IoT, automotive, mobile, industrial etc…)
- The state of security across devices is “variable”
  - “The S in IoT is for security” [Albert Einstein]
- Accessibility has improved
  - Proliferation of RISC-V
  - Open-source EDA
  - Affordable HW
- Supply Chain and trust
  - Geopolitical tensions have escalated the priority of hardware trust.
  - EU Cyber Resilience Act (CRA) incoming 2027!
- It is challenging, interesting and helpful!



https://audunsandaker.com/drawingsillustrations/hackerman2/

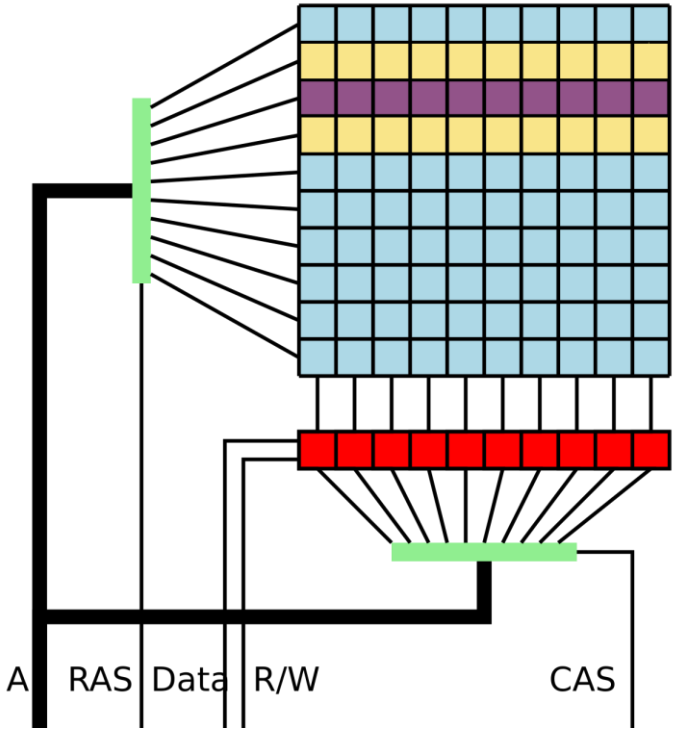
Examples of HW Exploits…

Rowhammer

- Repeated accesses of a row within a DRAM module generates electrical interference which can disturb the charge in an adjacent victim row, resulting in bit flips.
- As the density of DRAM cells increases, the problem gets worse. DRAM cells are “too close” to each other.
- Attack utilises the physics of the DRAM. No software bug is necessary.
- If unmitigated, bit flips can be performed predictably. Attacks have been constructed to perform privilege escalation.
- Variety of mitigations exist, but nothing is standardised.
- Great resources provided by Prof. Onur Mutlu

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

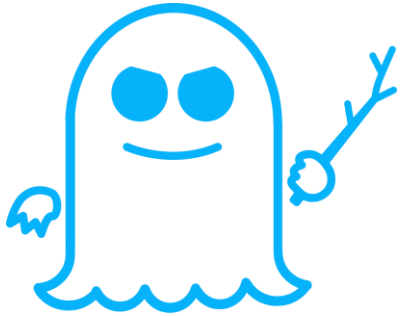
Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]
[1]Carnegie Mellon University [2]Intel Labs



https://en.wikipedia.org/wiki/Row_hammer

Spectre

- "*Spectre attacks involve inducing a victim to speculatively perform operations that would not occur during correct program execution and which leak the victim's confidential information via a side-channel to the adversary.*" [KO19]
- Micro-architectural hardware attack exploiting speculative execution and branch prediction.
- CPU discards the architectural state of miss-predicted speculations, but micro-architectural side-effects (e.g. cache state) persist.
- Hard to mitigate against as issue is in branch-predictor design.



https://en.wikipedia.org/wiki/Spectre_(security_vulnerability)

Spectre Attacks: Exploiting Speculative Execution

Paul Kocher[1], Jann Horn[2], Anders Fogh[3], Daniel Genkin[4],
Daniel Gruss[5], Werner Haas[6], Mike Hamburg[7], Moritz Lipp[5],
Stefan Mangard[5], Thomas Prescher[6], Michael Schwarz[5], Yuval Yarom[8]
[1] Independent (www.paulkocher.com), [2] Google Project Zero,
[3] G DATA Advanced Analytics, [4] University of Pennsylvania and University of Maryland,
[5] Graz University of Technology, [6] Cyberus Technology,
[7] Rambus, Cryptography Research Division, [8] University of Adelaide and Data61

Meltdown

- "*Meltdown exploits side effects of out-of-order execution on modern processors to read arbitrary kernel-memory locations including personal data and passwords.*" [LI20]
- Micro-architectural hardware attack exploiting Out-of-Order (OoO) execution.
- Similarly, CPU discards the architectural state, but micro-architectural side-effects (e.g. cache state) persist.
- Primarily mitigated using Kernel Page Table Isolation (KPTI).

https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability)

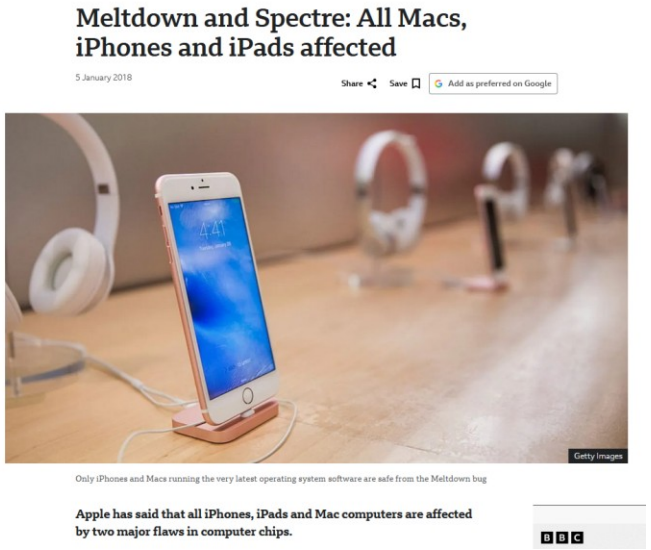
Meltdown: Reading Kernel Memory from User Space

Moritz Lipp[1], Michael Schwarz[1], Daniel Gruss[1], Thomas Prescher[2],
Werner Haas[2], Anders Fogh[3], Jann Horn[4], Stefan Mangard[1],
Paul Kocher[5], Daniel Genkin[6,9], Yuval Yarom[7], Mike Hamburg[8]
[1]*Graz University of Technology,* [2]*Cyberus Technology GmbH,*
[3]*G-Data Advanced Analytics,* [4]*Google Project Zero,*
[5]*Independent (www.paulkocher.com),* [6]*University of Michigan,*
[7]*University of Adelaide & Data61,* [8]*Rambus, Cryptography Research Division*

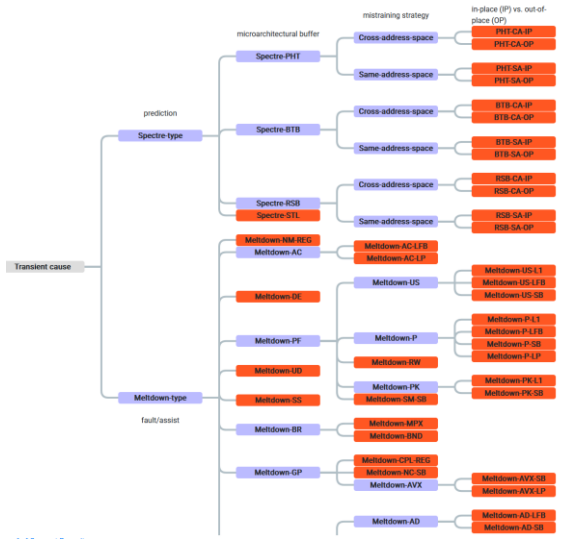
Tampere University

Spectre & Meltdown

- Both vulnerabilities were present in billions of deployed chips.
- SW patches were applied (KPTI, retpoline) were applied with severe performance penalties (5-30%)
- Root cause was in the microarchitectural design decisions, not the software.
- Many variants still being researched. Progress here: https://transient.fail/



https://www.bbc.com/news/technology-42575033

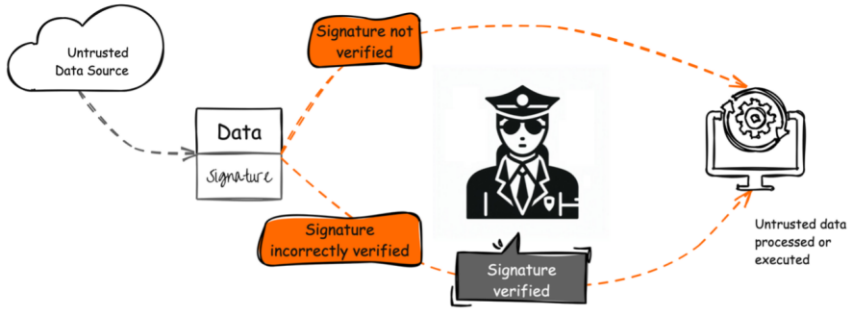


https://transient.fail/

EntrySign

- "*Improper signature verification in AMD CPU ROM microcode patch loader may allow an attacker with local administrator privilege to load malicious CPU microcode resulting in loss of confidentiality and integrity of a confidential guest running under AMD SEV-SNP.*" [cve-2024-56161]
- In some AMD Zen-based CPUs (Zen 1 – 4), it is possible to craft arbitrary/malicious microcode patches.
- CPU uses an insecure hash function in the signature validation of microcode updates.
- Mitigated using BIOS/UEFI updates.
- How did it get there in the first place??

https://bughunters.google.com/blog/zen-and-the-art-of-microcode-hacking

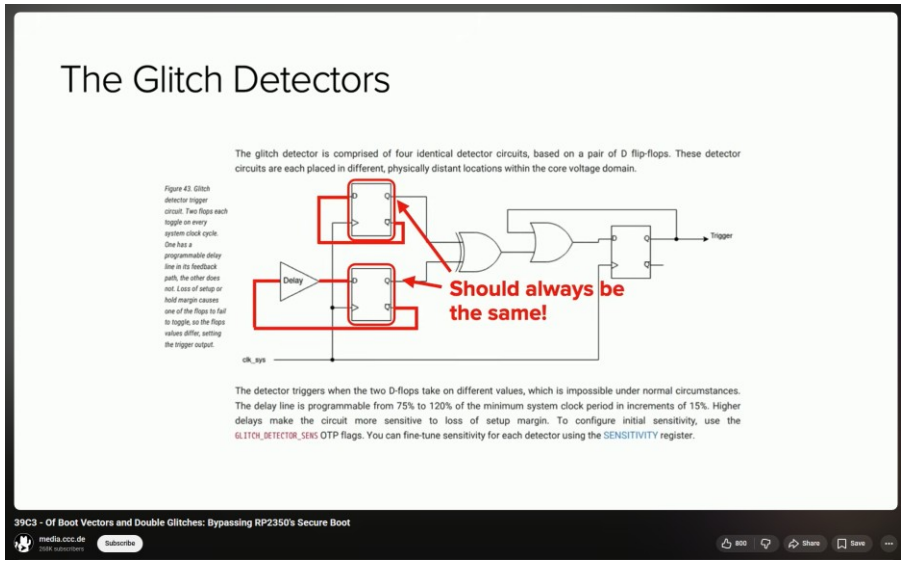


https://cwe.mitre.org/data/definitions/347.html

RP2350 Secure Boot

- RP2350 MCU was released in 2024.
- MCU contains many security features (and it is beautifully documented)
- Raspberry Pi launched a hacking challenge with a $20,000 prize (results here)
- Several hardware vulnerabilities identified.
- Great presentation detailing the process which was used to find two of the vulnerabilities: **39C3 - Of Boot Vectors and Double Glitches: Bypassing RP2350's Secure Boot**
- **Security through transparency** ❤️

https://www.raspberrypi.com/products/rp2350/



https://www.youtube.com/watch?v=V5KvW4elzXU

Basic Security Concepts

Security Objectives

- Well-known security properties of code or data which can be used to secure a system [SZ20] (“CIA Triad” or “Cornerstone Security Properties”):

1. **Confidentiality**: The prevention or disclosure of secret or sensitive information to unauthorised users or entities.
2. **Integrity**: The prevention of unauthorised modification of protected information without detection.
3. **Availability**: The provision of services and systems to legitimate users when requested or needed.

… but also important for us is:

Authenticity: relates to determining who a User or System is. Inherently requires integrity as an attacker should neither be able to modify the authentication information nor make up their own [LE13].

- Note that other security properties exist.

https://preview.redd.it/when-will-tegridy-finally-end-v0-3uze0jnb9luc1.jpeg?auto=webp&s=a1cbe102da202f507807397b1f5ffe22aa584daa

Kerckhoff's Principle

- Can be paraphrased as:

  "The operations of a secure system should be publicly known and should have no secrets other than the cryptographic keys. Thus, if an attacker knows everything about the operation of the system, they still cannot break the system without the cryptographic keys"

- Designers should not underestimate the cleverness of attackers!

- **Security through obscurity doesn't work** and has led to real life attacks:

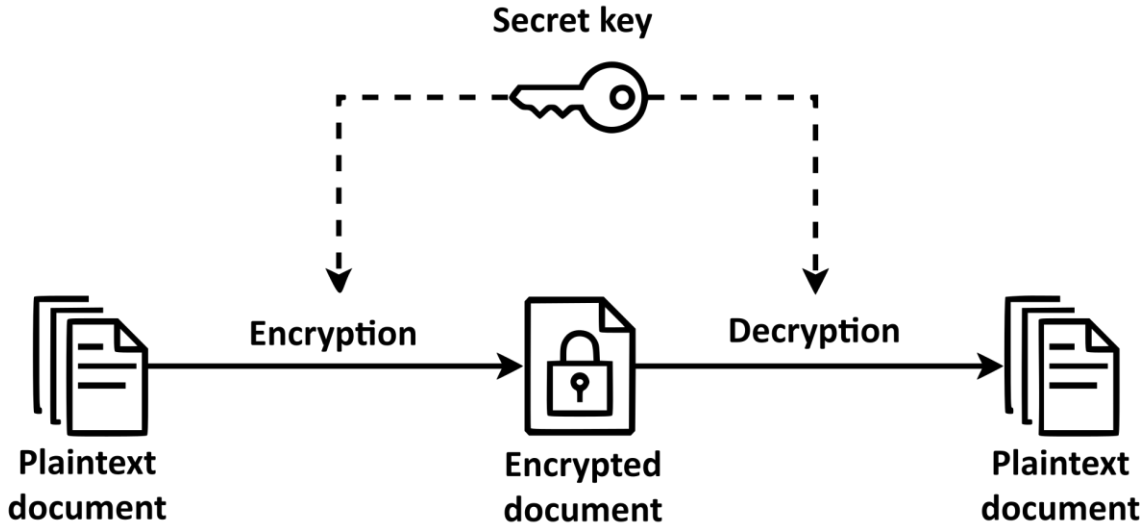
  Chips used within London's Oyster cards (MIFARE classic chips) used a weak, proprietary, 48-bit cipher known as 'Crypto-1'. The algorithm was secret, but reverse-engineered by academics and shown to have severe vulnerabilities. Once broken, the cards could be cloned in less than a second using a passive, low-cost wireless reader. [CO09]

https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle

https://en.wikipedia.org/wiki/Oyster_card

Cryptography Basics - Symmetric

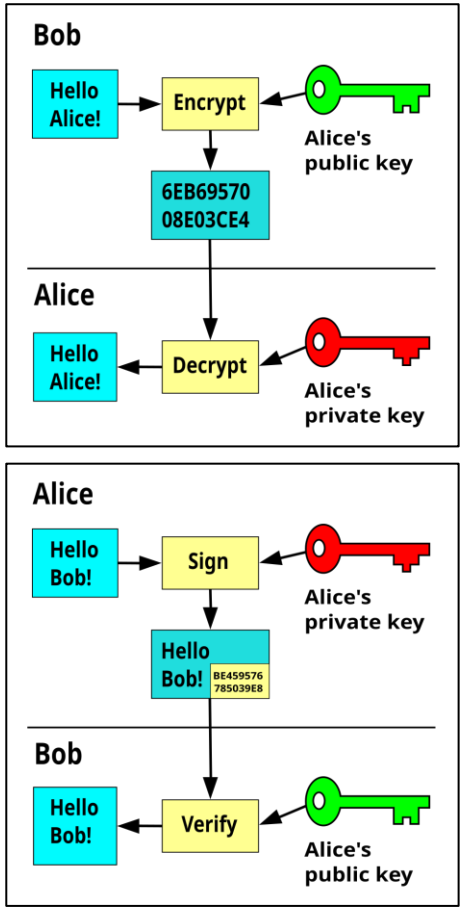
- Symmetric cryptography (private-key cryptography)
- Data encryption and decryption uses the same secret key.
- Ciphers come in block and stream algorithms.
- Often used for bulk-data encryption as faster than alternatives.
- Examples include AES, DES, SM4, ChaCha.



https://en.wikipedia.org/wiki/Symmetric-key_algorithm

Cryptography Basics - Asymmetric

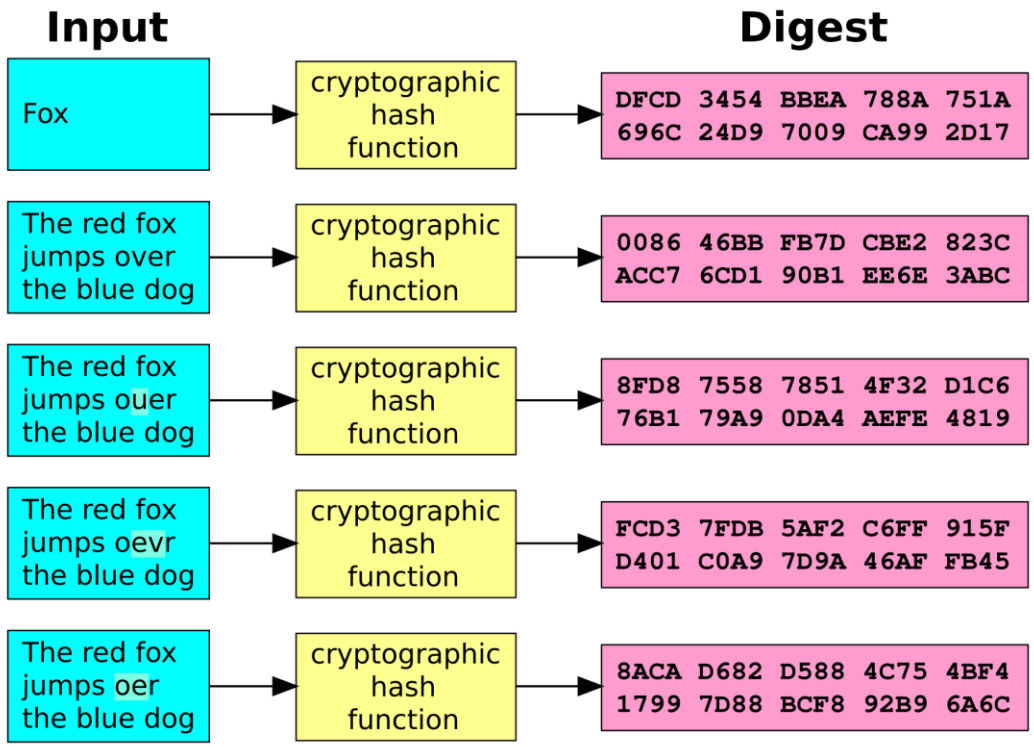
- Asymmetric cryptography (public-key cryptography)
- Data encryption and decryption use different keys.
  - Public key and private (secret) key
- Public key is used to encrypt data and private key is used to decrypt data.
  - Almost reversed when used with digital signatures.
- Public key can be given to anybody to encrypt data, but only the owner of the private key can read the encrypted data (thwarts man-in-the-middle attacks)
- Significantly slower than symmetric cryptography.
- Used for key-encapsulation and signature verification.
- Examples include RSA, DSA, ECDSA.



https://en.wikipedia.org/wiki/Public-key_cryptography

Cryptography Basics – Hash Functions

- Cryptographic hash functions
- Maps input data of variable size to a fixed sized output.
- One-way function - infeasible to mathematically reverse the operation or deduce input data from output.
- Output of a hash function sometimes referred to digest or fingerprint.
- Commonly used for authentication and integrity checking e.g. message authentication codes (MACs) and digital signatures.
- Examples include SHA-2/3.



https://en.wikipedia.org/wiki/Cryptographic_hash_function

Post-Quantum Cryptography

- Most modern asymmetric (public-key) cryptography derives its security from the computational difficulty of Integer factorisation and discrete logarithms. A sufficiently power quantum computer could theoretically solve these problems using Shor's algorithm.
- Symmetric (private-key) cryptography and hash functions are unaffected by this, but Grover's algorithm can offer a quadratic increase in search time (reducing AES-128 to 64b of effective security). [SZ20]
- Post-Quantum Algorithms are those which are not vulnerable to the above.
- NIST has recommended that all vulnerable algorithms are to be considered as deprecated by 2030 and disallowed after 2035[1].
- New NIST standardised PQC algorithms:
  - ML-KEM "Kyber" – For Key establishment
  - ML-DSA "Dilithium", SLH-DSA "SPHINCS+" and FN-DSA "FALCON" - For Digital Signatures.
- The transition to PQC affects many applications and efficient/secure implementations are still being explored!

FALCON

Fast-Fourier Lattice-based Compact Signatures over NTRU

https://falcon-sign.info/

https://pq-crystals.org/kyber/index.shtml

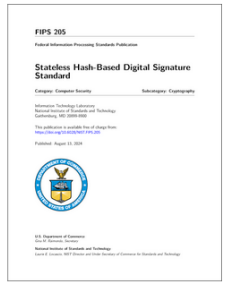
https://pq-crystals.org/dilithium/

FIPS 203

Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)

FIPS 204

Module-Lattice-Based Digital Signature Standard (ML-DSA)



FIPS 205

Stateless Hash-Based Digital Signature Standard (SLH-DSA)

https://csrc.nist.gov/projects/post-quantum-cryptography

1. https://doi.org/10.6028/NIST.IR.8547.ipd

Tampere University

Side-Channels

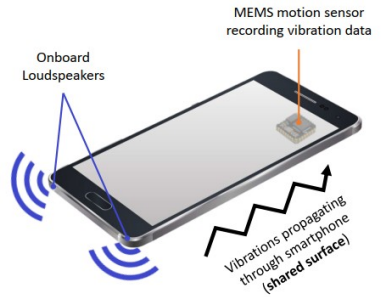
- A side channel is a communication channel through which the sender did not intend to send information to the receiver.
- The data is not sent over the channel, but rather ‘leaked’ as a side-effect of the selected communication channel implementation.
- Leakage could be through electro-magnetic (EM) emanations, acoustic emanations, timing, power etc…

SoK: Assessing the Threat Potential of Vibration-based Attacks against Live Speech using Mobile Sensors

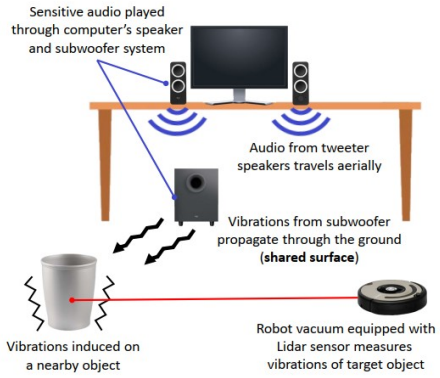
Payton Walker
University of Alabama at Birmingham
Birmingham, Alabama, USA
prw0007@uab.edu

Nitesh Saxena
University of Alabama at Birmingham
Birmingham, Alabama, USA
saxena@uab.edu

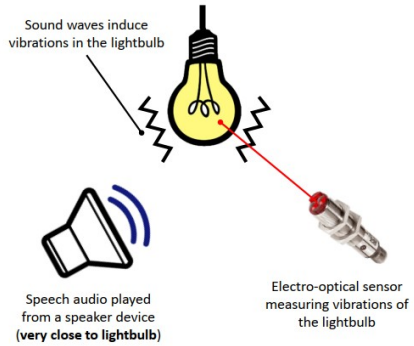
(a) Setup used in Gyrophone [42]



(b) Setup used in AccelEve [13] and Kinetic Song Recognition [41]



(c) Setup used in Lidarphone [54]



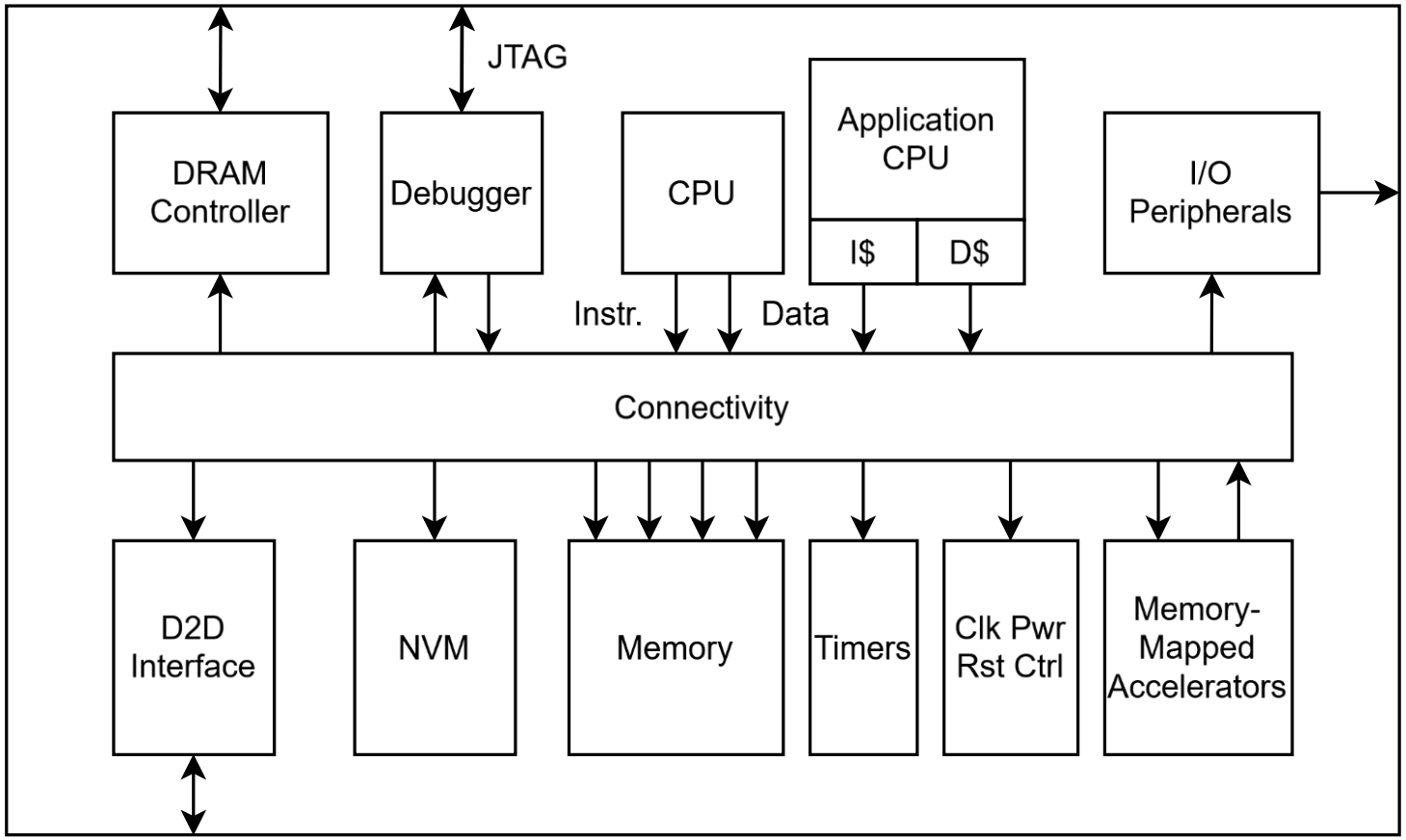
(d) Setup used in Lamphone [45]

[WS21]

SoC Architecture

Recap for context

General SoC Architecture

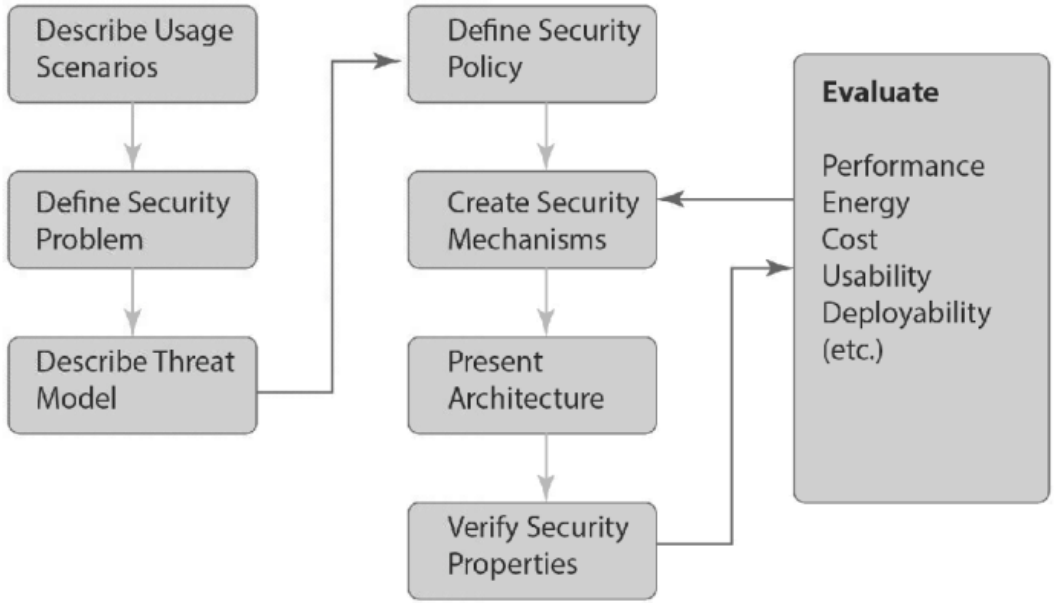


Threat-Based Design

How do we determine attack vectors and required mitigations?

Threat Modelling

- When designing a chip, a proactive security methodology should be applied such that security requirements are formed during the definition phase and not as an afterthought.
- To define what mitigations/defences are required, the threats to the system must be systematically identified and defined.
- Standardised certification methodologies/frameworks exist (e.g. ISO/ISEC 15408 Common Criteria) and require threat modelling.
- General procedure might look like [LE13]:
  - Establish the “security context” and assets
    - Define the intended environment for the SoC, identify assets requiring protection and the trust boundaries of the system. This must be done for the full SoC life-cycle.
  - List attack surfaces
    - Classify interfaces by attacker capability required: remote (no physical access), local (software on-device), and physical (requires hands-on hardware access)

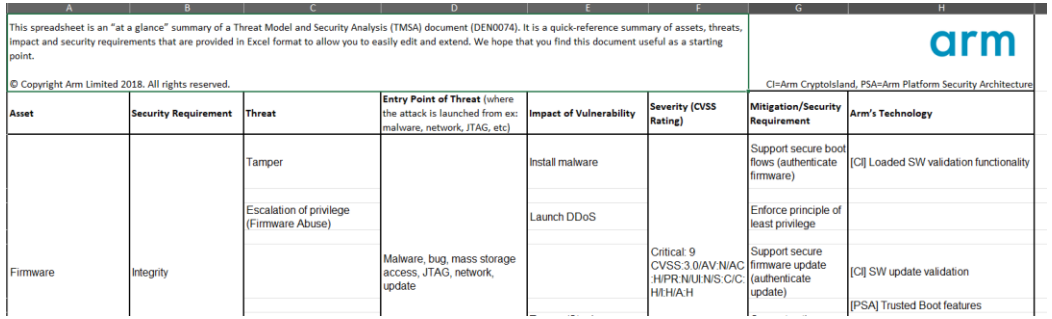


Security architecture design methodology [LE13]

Tampere University

Threat Modelling

- Assess the attacks:
  - Calculate the “attack potential”
    - Score threats using attacker capability required, tooling costs, time spent, equipment etc…
    - Impact should consider confidentiality, integrity, availability and authentication.
    - Threats which are not considered should be explicitly listed and justified.
  - Establish the mitigations
    - Map each of the threats to a concrete hardware/firmware countermeasure.
    - Few mitigations are perfect.
- Outputs of this process will be security requirements and properties which will inform the architecture and design of the SoC.
- Threat models and associate artefacts are living documents throughout development process.



This spreadsheet is an “at a glance” summary of a Threat Model and Security Analysis (TMSA) document (DEN0074). It is a quick-reference summary of assets, threats, impact and security requirements that are provided in Excel format to allow you to easily edit and extend. We hope that you find this document useful as a starting point.

arm

CI=Arm Cryptoisland, PSA=Arm Platform Security Architecture

Asset	Security Requirement	Threat	Entry Point of Threat (where the attack is launched from ex: malware, network, JTAG, etc)	Impact of Vulnerability	Severity (CVSS Rating)	Mitigation/Security Requirement	Arm's Technology
Firmware	Integrity	Tamper	Malware, bug, mass storage access, JTAG, network, update	Install malware	Critical: 9 CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H	Support secure boot flows (authenticate firmware)	[CI] Loaded SW validation functionality
		Escalation of privilege (Firmware Abuse)		Launch DDoS		Enforce principle of least privilege	
						Support secure firmware update (authenticate update)	[CI] SW update validation
							[PSA] Trusted Boot features

ARM Water Meter Threat Model DEN0074

Asset category	Asset	Security property	Attack profile	Attack path	Mitigation
Fuse/OTP high value secrets	UDS Seed	Confidentiality and integrity	Expert	Malicious manufacturing spoofing of UDS Seeds	UDS obfuscation with class RTL key
Fuse/OTP high value secrets	UDS Seed	Confidentiality and integrity	Expert	Invasive attack (fuse analysis)	Shield fuse IP
Fuse/OTP high value secrets	UDS Seed	Confidentiality and integrity	Expert	Boot path tampering while retrieving UDS values	UDS obfuscation with class RTL key
Fuse/OTP high value secrets	UDS Seed	Confidentiality and integrity	Expert	Attempting to derive die specific keys by knowing UDS	Confine unobfuscated UDS and subsequent derivations to key vault
Fuse/OTP high value secrets	Field entropy	Confidentiality and integrity	Expert	Malicious manufacturing spoofing on field entropy	Field entropy obfuscation with class RTL key
Fuse/OTP high value secrets	Field entropy	Confidentiality and integrity	Expert	Invasive attack (fuse analysis)	Shield fuse IP

https://github.com/chipsalliance/Caliptra/blob/main/doc/Caliptra.md

Attack Surfaces

Side-Channel Attacks

- **Power Analysis**
  - Measurement of supply current during execution of crypto operations can be used to extract keys statistically (differential power analysis - DPA) or by observing data-dependent variations in power consumption (simple power analysis (SPA)).
  - Excellent paper explaining how this works [KO11].
- **Electromagnetic**
  - Similar to power analysis, but no physical contact is required. Near-field EM probes can be used to target specific blocks on the die.
- **Timing Attacks**
  - Execution-time variation leaks data-dependant branching information.
  - Cache side-channel attacks are a form of timing side-channel attack.

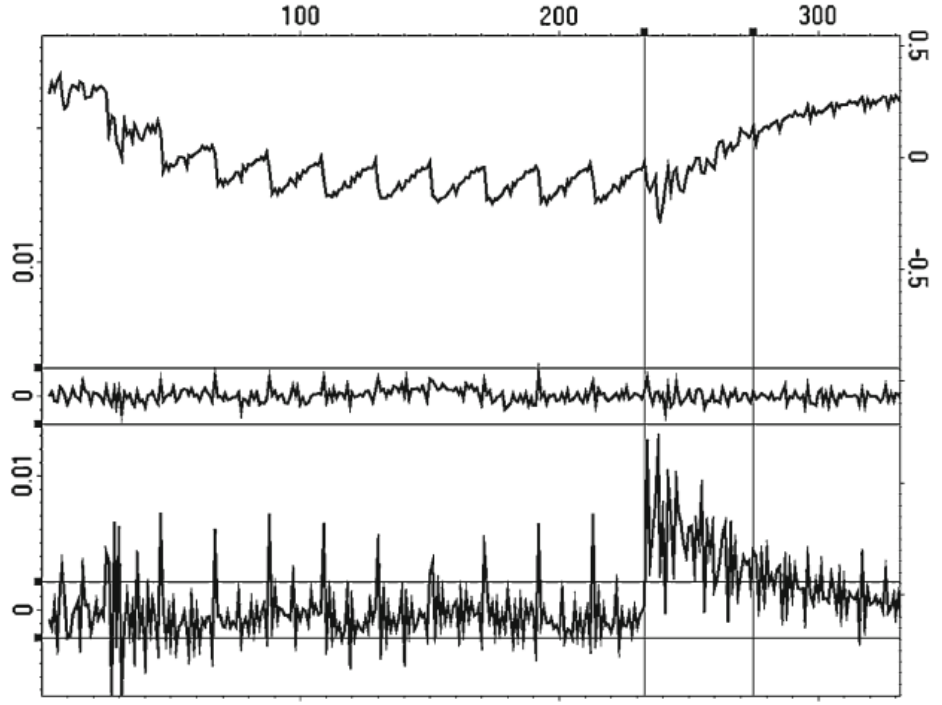


Fig. 9 DPA results showing the average trace for an AES-128 operation running on an FPGA (top), the differential trace for an incorrect guess of a byte of the last round key (middle) and the differential trace for the correct key byte (bottom)
[KO11]

Fault Injection Attacks (FIA)

- Voltage glitching
  - Briefly drop supply voltage to cause timing errors in logic. Can cause CPU to skip an instruction or produce corrupted outputs.
- Clock Glitching
  - Inject additional pulses to cause setup-time violations. CPU executes instructions incompletely and effectively skips it or corrupts result.
- Laser Fault Injection
  - Target a specific gate or memory cell with a focused laser pulse. Can cause a single bit-flip. Requires de-capping chip.

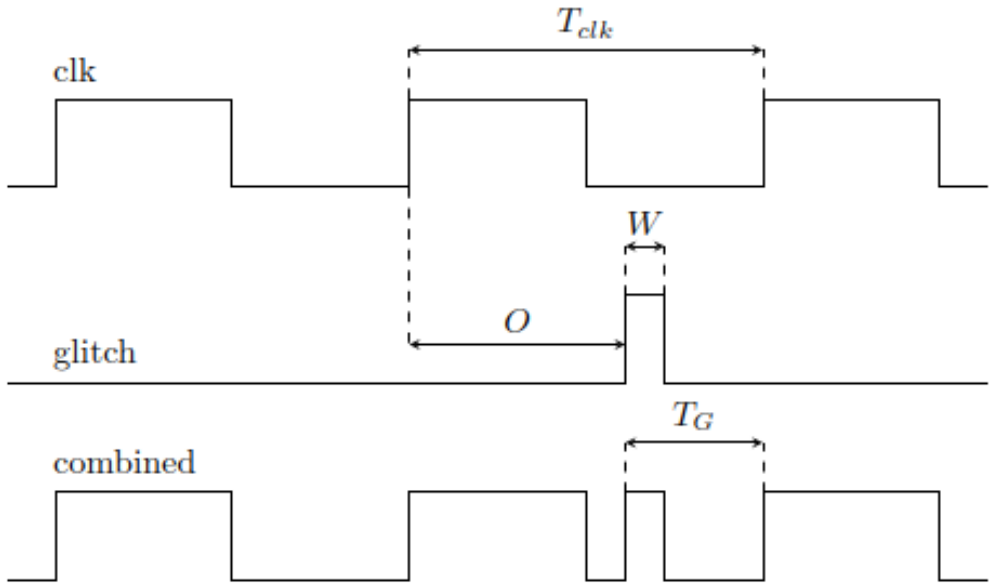
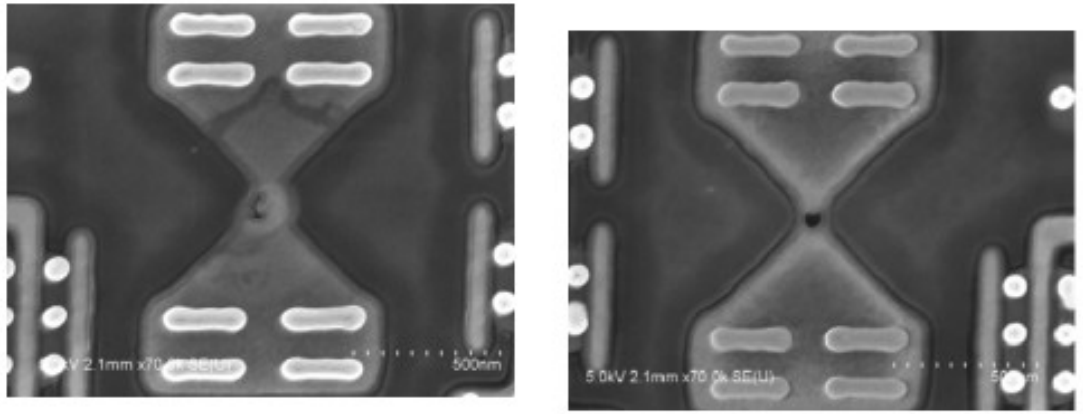


Figure 4: Glitch inserted in the low part of a clock period.

[AS23]

JTAG Port Access

- Often debug ports are disabled on chips following commissioning.
  - Sometimes done by blowing fuses on the chip (not always effective – see figure)
  - Sometimes they are not disabled at all…
- RISC-V external debug spec. defines that the Debug Transport Module (DTM) gives full access (M-Mode).
- Design For Test (DFT) logic typically exposed via JTAG and can be used to exploit Automatic Test Pattern Generation (ATPG) logic to extract information. [YA24]



(a) Fuse incorrectly blown (i.e. not completely open).

(b) Fuse properly blown.

Peeters, E. (2015) 'SoC security architecture: current practices and emerging needs', in *2015 52nd ACM/EDAC/IEEE Design Automation Conference, DAC 2015*. [Online]. 1 June 2015 New York, NY, USA: ACM. p.

Cold Boot/Memory Remanence Attacks

- **Cold Boot**
  - Cold Boot attacks exploit the fact that charge decay in capacitors slows down at lower temperatures.
  - After a system is powered off, DRAM can be cooled (e.g. using off-the-shelf compressed air cans) and transferred to an attacker's machine.
  - Any sensitive, unencrypted data held in DRAM memory can be extracted and recovered [YI17].
- **Memory Remanence**
  - Similar to Cold Boot attacks but applies more to other memory technologies also (SRAM and flash).
  - It has been shown that flash memory can be recovered after erase operations [SK05].

Figure 2: **Cold Boot Attack on DDR4 DRAM.** This photo shows the DRAM in one of our DDR4-based systems. The DRAM is filled with data scrambled by the memory interface of an Intel Skylake-based CPU. The memory has been cooled to $-25°C$, and it will next be moved to a separate system where its contents will be descrambled.

[YI17]

Supply Chain Attacks

- **Hardware trojans**
  - Malicious logic inserted into an IP block during fabrication. Can lie dormant, activating only when specific conditions are met.
  - Difficult to detect. Standard testing only checks functional correctness and not adversarial behaviour.
- **Counterfeit/Tampered Chips**
  - Recycled or re-marked chips sold as new with modifications applied (e.g. FW modifications within mask ROM)

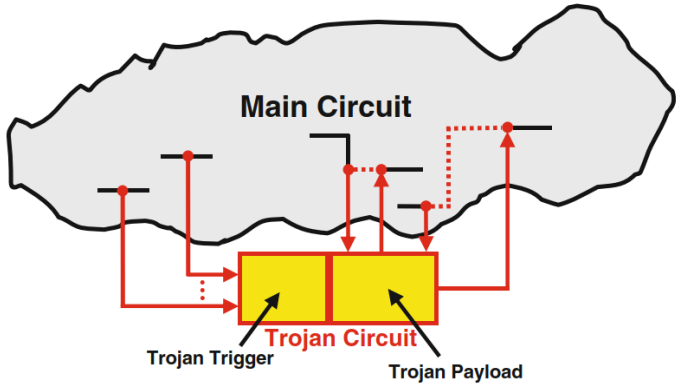


Fig. 1.2 Functional Trojan implementation

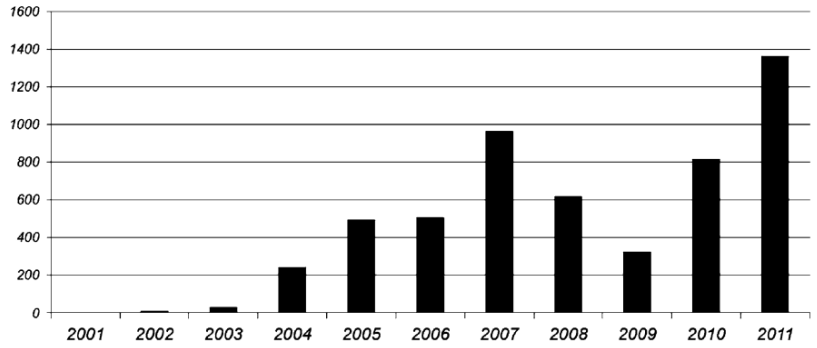


Fig. 1.6 Counterfeit incidents reported by IHS [29]

[TE14]

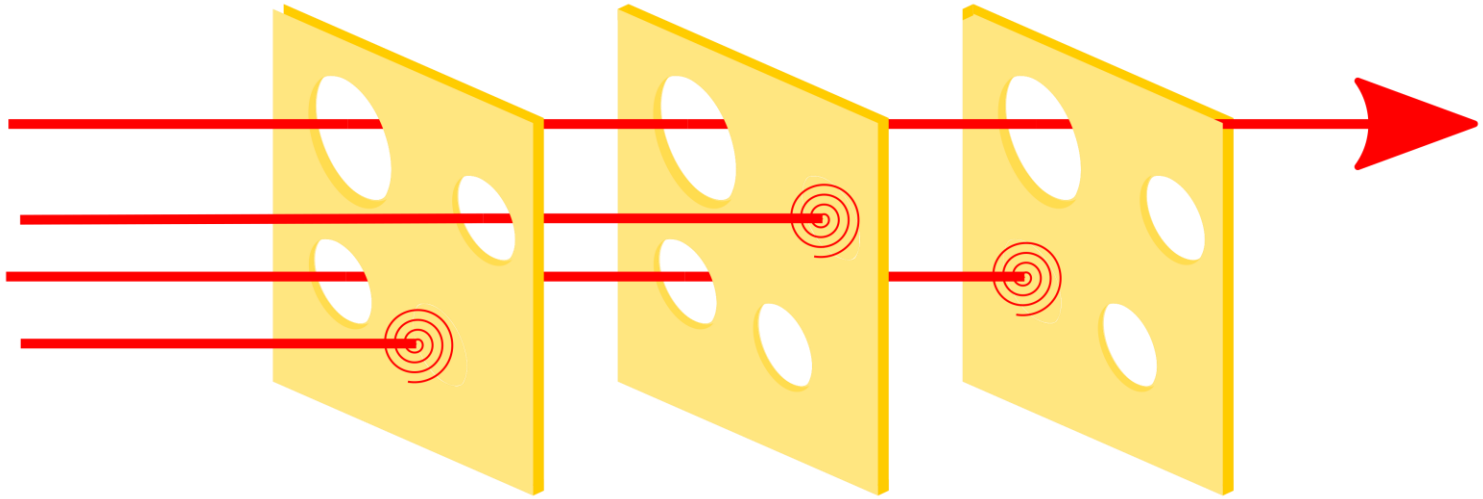
There are many more we know of…

…and like many that we don’t know of YET

Mitigations

Hardware Countermeasures

- Rarely a 1:1 mapping between threats and countermeasures.
- A multi-layered approach is often required to effectively thwart attacks.
- Think of the “Swiss cheese model”
- Some examples…



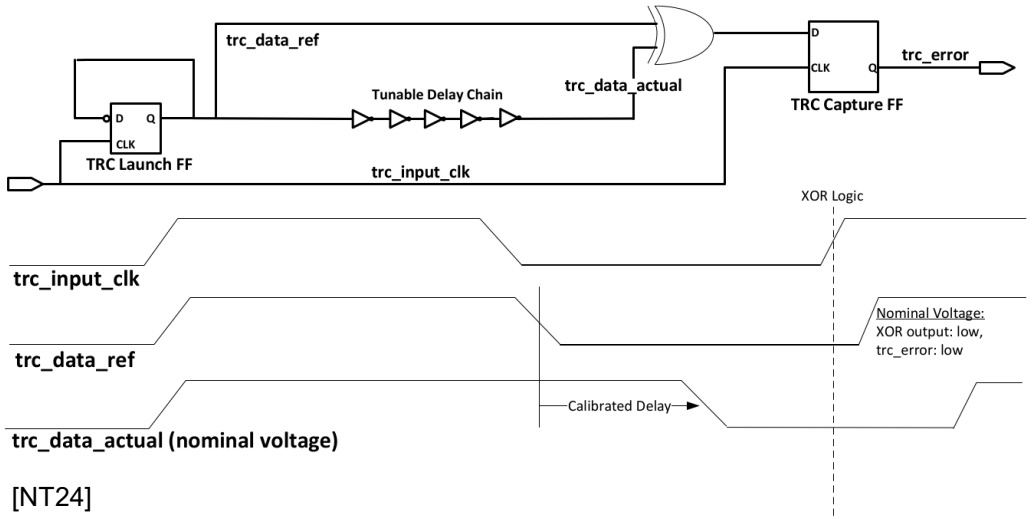
https://en.wikipedia.org/wiki/Swiss_cheese_model

Some Architectural Hardware Countermeasures

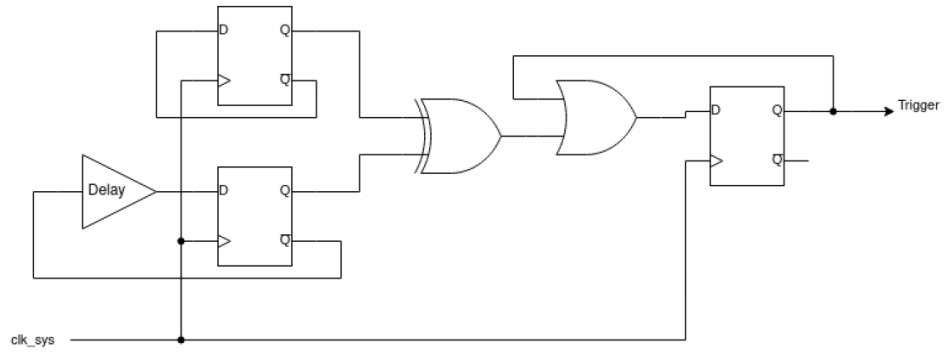
Tampere University

Glitch Detectors

- Sensors for detecting voltage and clock fault injection attacks.
- The idea is that if an attack is detected, the system can take appropriate e.g. reset the system or prevent the release of potentially faulty data.
- Utilises the known physical delay properties of the gates to detect abnormal timing conditions (maybe think about how aging of the silicon affects this…).
- Interesting review of glitch detector effectiveness in [AS23] .



[NT24]



https://datasheets.raspberrypi.com/rp2350/rp2350-datasheet.pdf

Tamper Detection

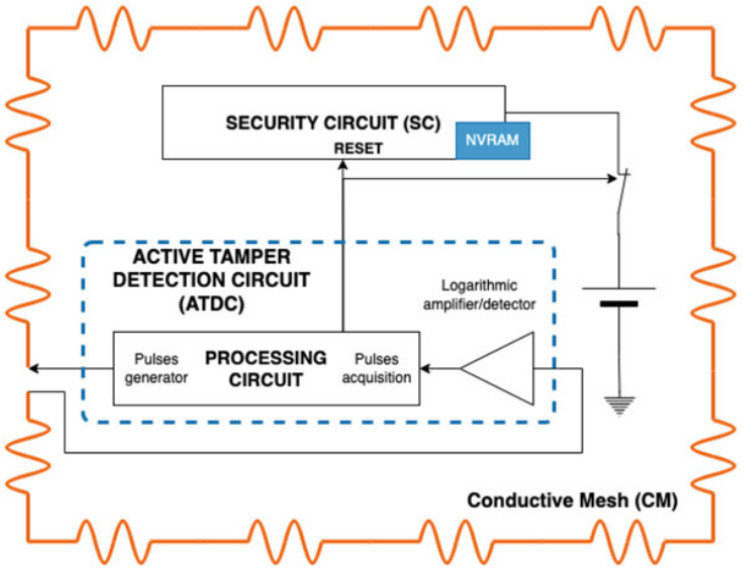


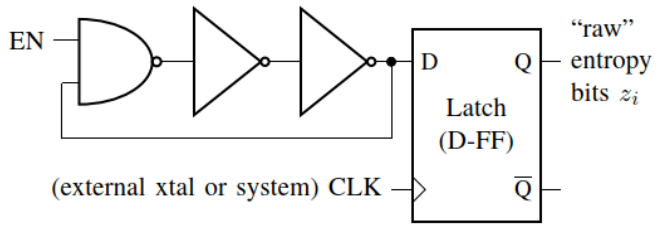
Fig. 15.8 Diagram of the security system, enclosed by the conductive mesh

Tehranipoor, M. et al. (2023) *Hardware security primitives*. 1st edition. [Online]. Cham: Springer.

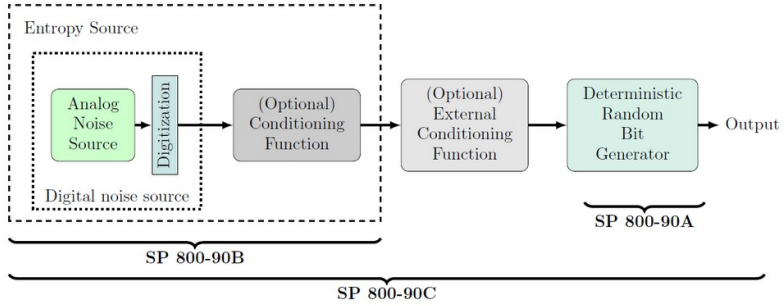
- Many attack vectors require physical access to the silicon.
- Various mechanisms can be employed to detect or protect from physical alterations to the package or circuitry:
  - Active anti-tamper meshes,
  - Laser-absorbing shields,
  - Temperature-sensitive circuits,
  - Photonic Sensors,
  - Aging Detection circuits.

Random Number Generators (RNG)

- “Randomness” is a key requirement for the seed material and initialisation vectors used within cryptographic operations.
- It must be truly random and therefore cannot rely on arithmetic methods.
- Thermal jitter (phase noise) from a free-running ring oscillator is a common, easily implementable physical randomness source which is used within many True Random Number Generators (TRNGs). [SA21]
- Entropy sources are regulated within cryptographic standards such as NIST SP 800-90B. In addition to the TRNG, they also mandate health-monitoring and post-processing.
- A TRNG is slow and so can be used to seed a (much faster) Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) to generate entropy.



[SA21]



https://csrc.nist.gov/Projects/random-bit-generation

Cryptographic Acceleration

- **Instruction Set Extension (ISE)** - based Acceleration
  - Processor functional units are extended to support specialised instructions used for cryptographic operations.
  - More efficient than software-based implementations and offer better side-channel resistance [MS25]
  - Intel, Arm and RISC-V ISAs all include cryptographic extensions (shameless self-promotion: [SZ42]).
- **Discrete Accelerators**
  - Dedicated IP which typically uses a register-based control interface to perform cryptographic operations.
  - Can be more performant, but less flexible than software and ISE-based
- Secure key management in both needs careful consideration!

[SZ24]

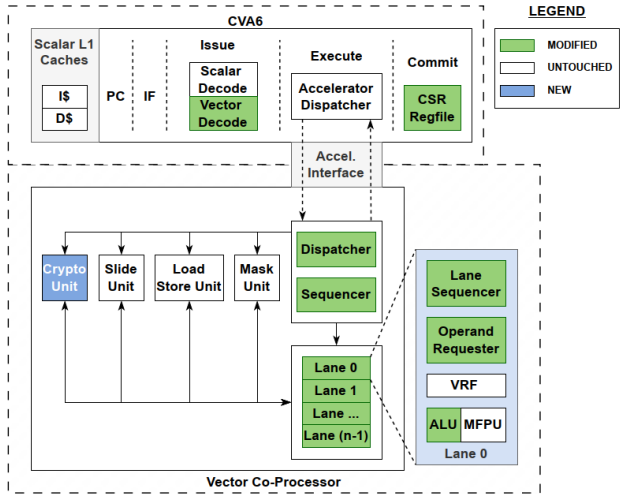
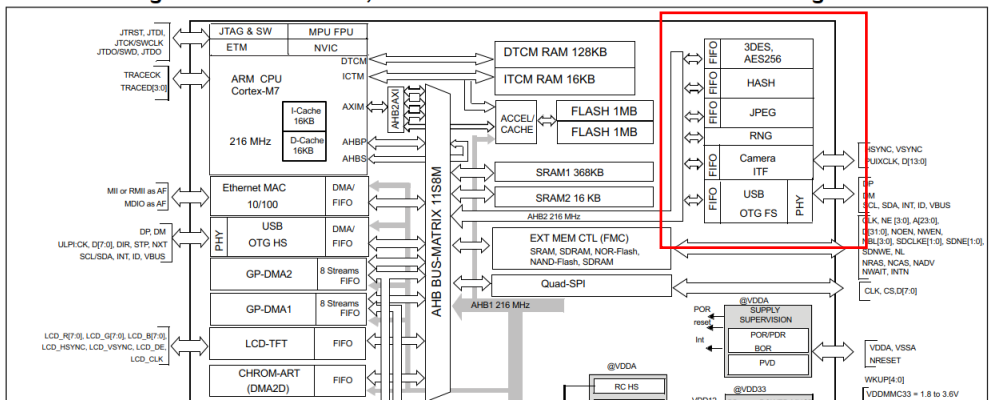


Figure 2. STM32F777xx, STM32F778Ax and STM32F779xx block diagram



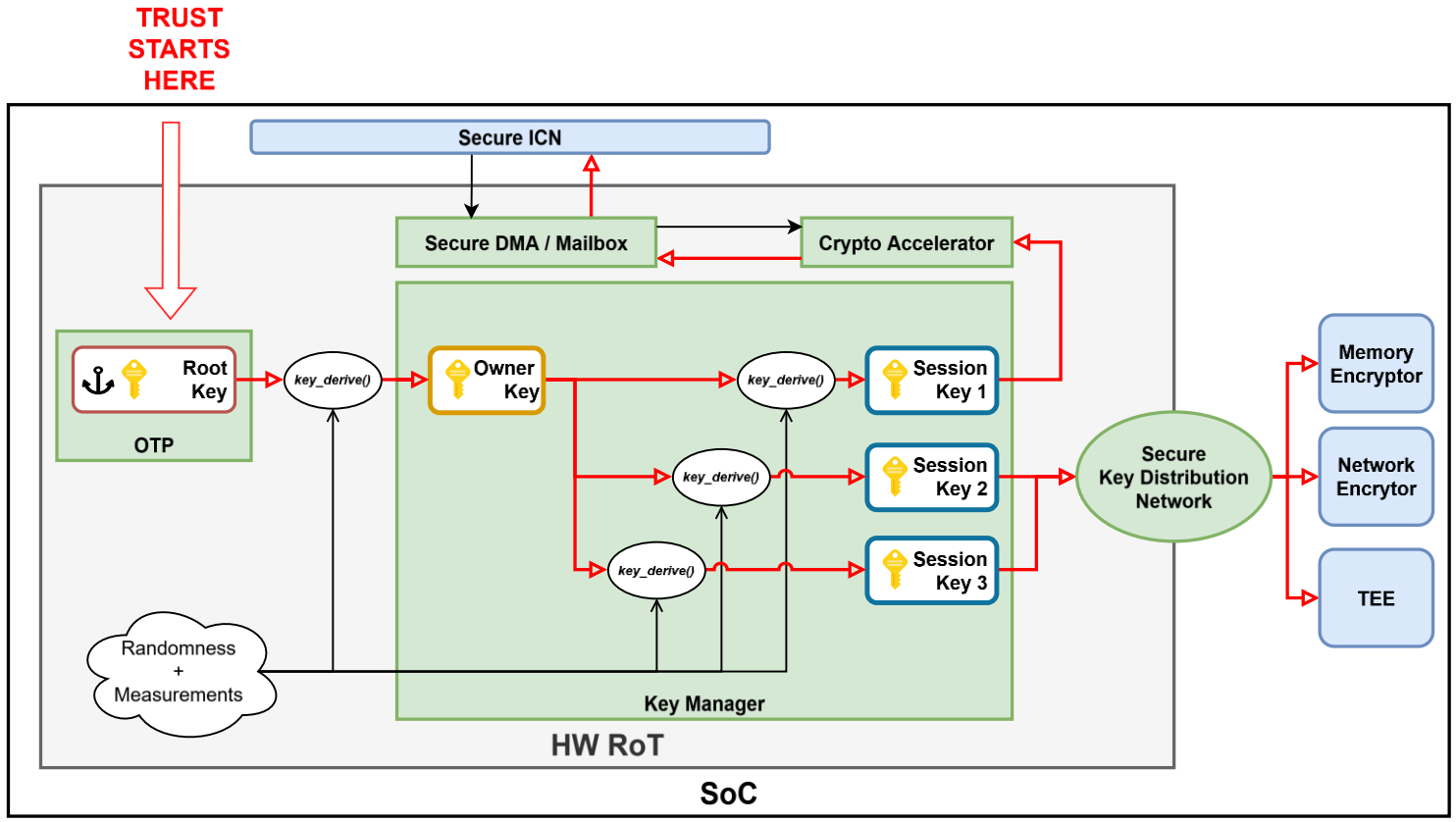
STM32F777xx Datasheet - DocID028294 Rev 3

Hardware Root-of-Trust (HW RoT)

- Recall that secure subsystems need to be able to offer ***at least***, confidentiality, integrity, availability, and authentication properties to operate securely and be trusted:
  - **Confidentiality**: Subsystems need encryption keys for confidentiality of data leaving the subsystem.
  - **Integrity**: Subsystems need to applies keys for integrity checking of data leaving the processor e.g. hash function within a MAC.
  - **Availability:** Critical subsystem functionality must be monitored to ensure that there is no denial of service.
  - **Authentication**: Subsystem must authenticate itself using keys to convince a remote party that data/messages are coming from a trusted system.
- The keys used by the system are integral to the trustworthiness/security of the system.
- If you can't trust the hardware you are running on, no amount of software security can save you.
- Overall trust is derived from...
  - A cryptographic key or set of cryptographic keys,
  - Trust in the manufacturer (supply chain),
  - The immutable properties of the hardware itself.

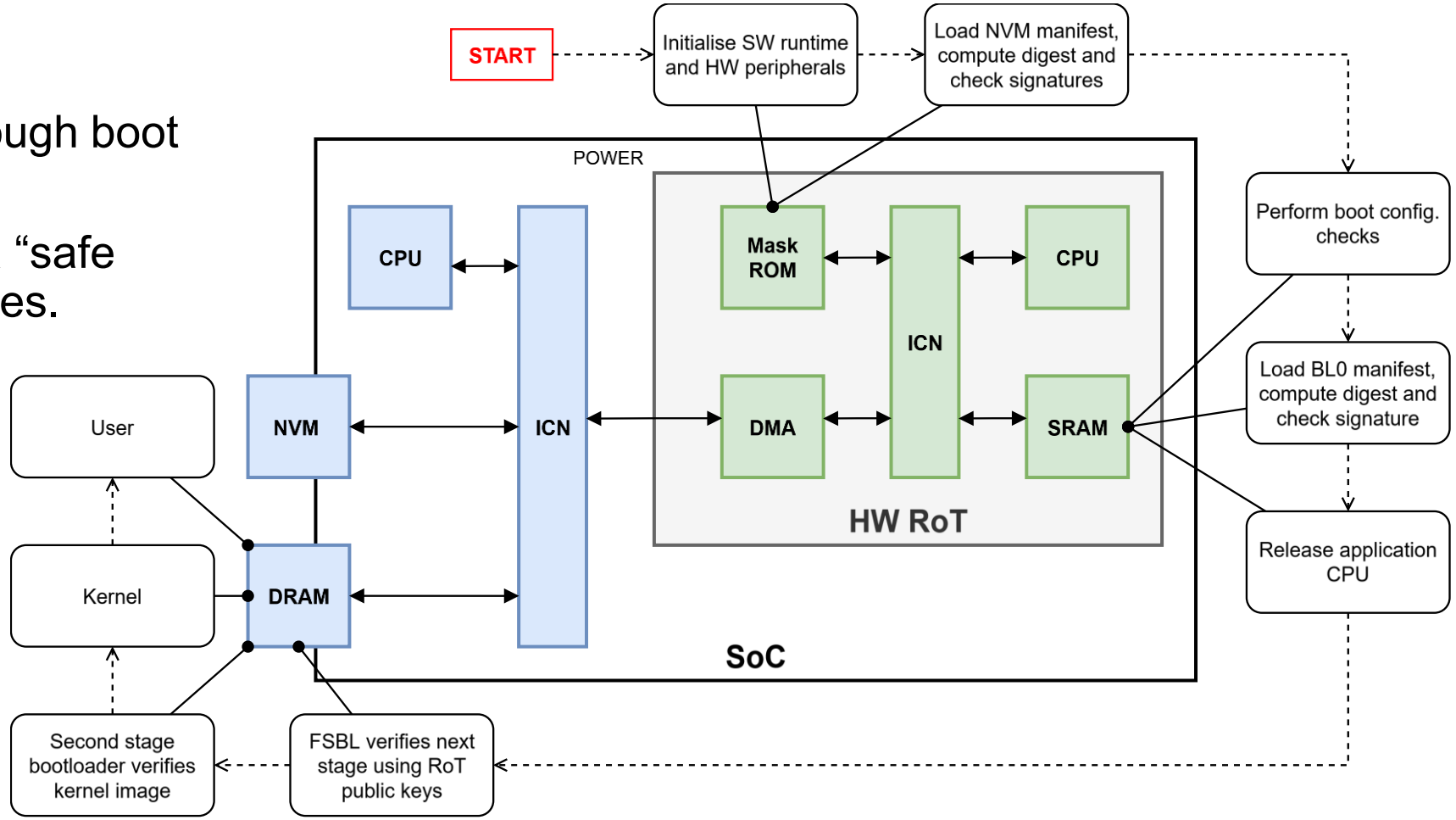
Hardware Root-of-Trust

- The hardware Root-of-Trust (HW RoT) is a set of trusted hardware components from which all security guarantees of a system are derived.
- It forms the cryptographic foundation of the device and offers a physically immutable trust anchor for each device.



Hardware Root-of-Trust

- Secure Boot
- Trust propagates through boot components.
- RoT will fall-back to a “safe state” if any stage failes.



Hardware Root-of-Trust

- Typical functions performed by HW RoT:
  - Secure Boot,
  - Secure key storage and generation,
  - Remote attestation,
  - Secure update,
  - Life-cycle Management
  - Cryptographic acceleration,
  - Monitoring of security functions.
- Typical components forming a HW RoT:
  - Immutable boot ROM – Cannot be modified post-manufacture and therefore is inherently trustworthy,
  - Secure key storage – Stored within OTP, secure registers or utilising a PUF,
  - Cryptographic accelerators,
  - Isolation and secure interfaces e.g. secure mailbox,
  - Random number generators,
  - Secure peripherals.

Physically Unclonable Functions (PUFs)

- A PUF is a hardware-based security primitives which leverage process variations in the chip to create a unique device identifier.
  - Resembles biometric data of people e.g. a fingerprint.
- Unclonable, tamper-evident and can be used to generate keys.
- Eliminates need to store keys in NVM, therefore reducing the risk of key exposure.
  - Device-specific secret data (secret keys) is performed during manufacture. This introduces a risk as the manufacturer has access to device secrets.
  - Can be combined with the HW RoT to generate the device secrets.
- Various constructions e.g. arbiter, glitch, SRAM, ring oscillator etc...
- Very active research space.

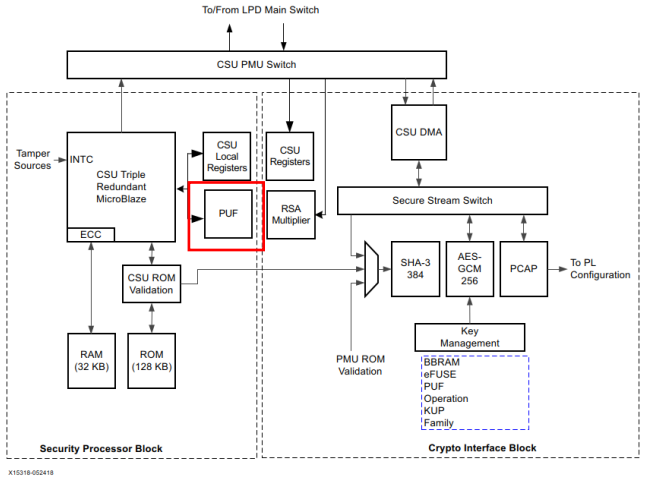
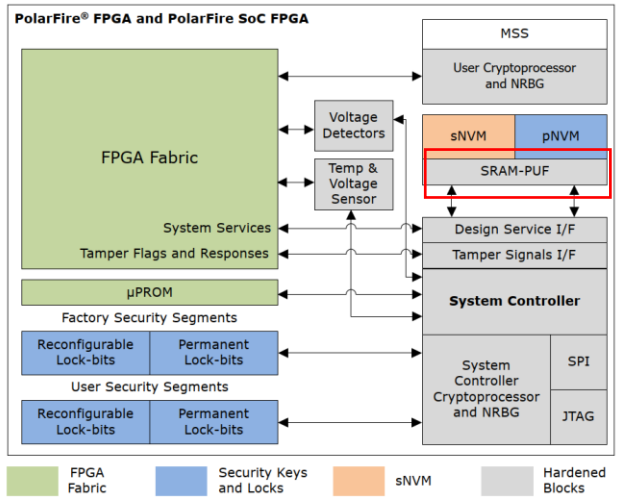


Figure 12-1: **Configuration Security Unit Block Diagram**

Zynq Ultrascale+ Device TRM UG1085 (v2.2)



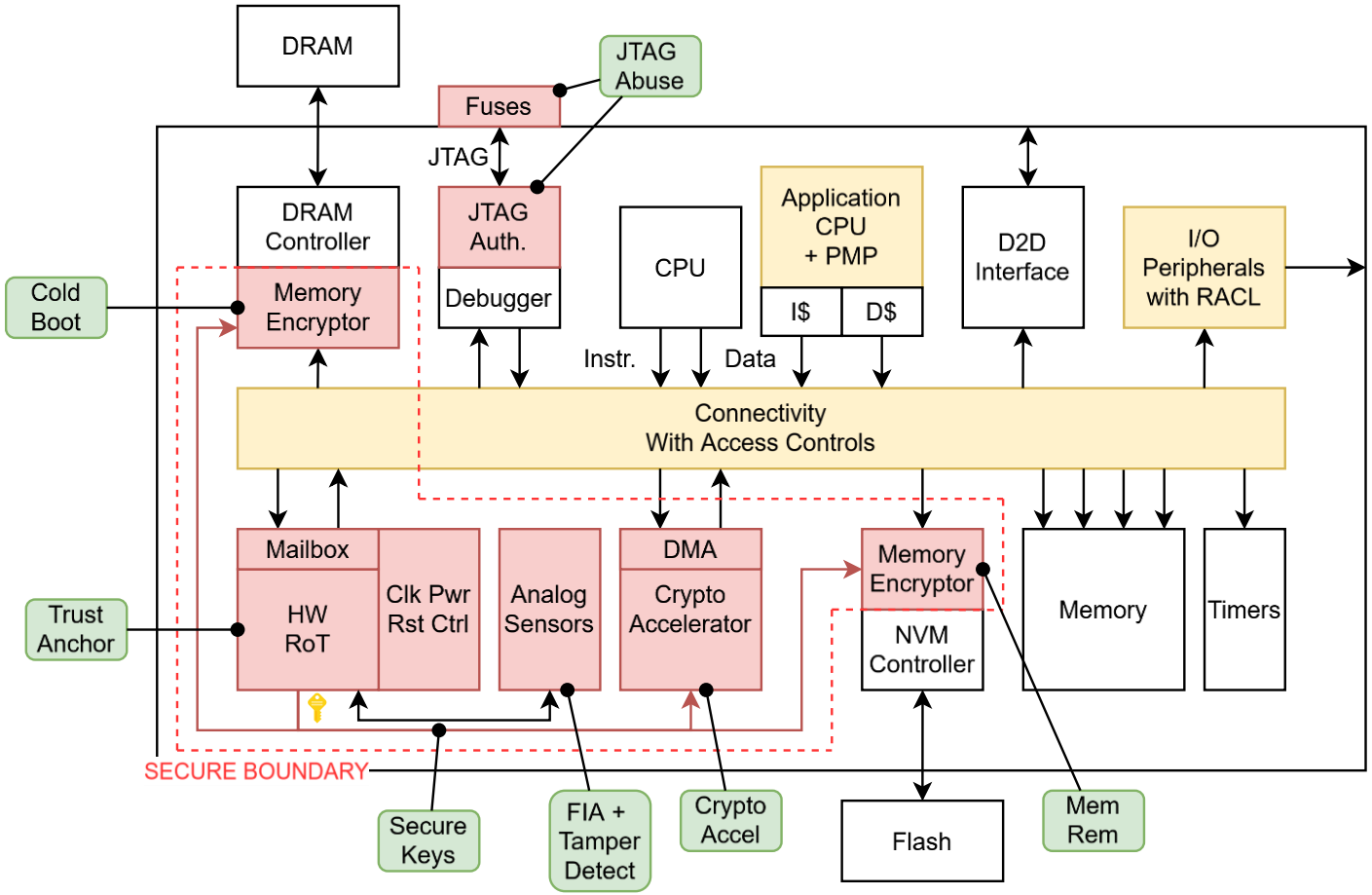
Polarfire Family FPGA Security User Guide DS60001726J

Again…there are many more…

…This topic is HUGE

Let’s Update the Architecture…

Updated SoC Architecture

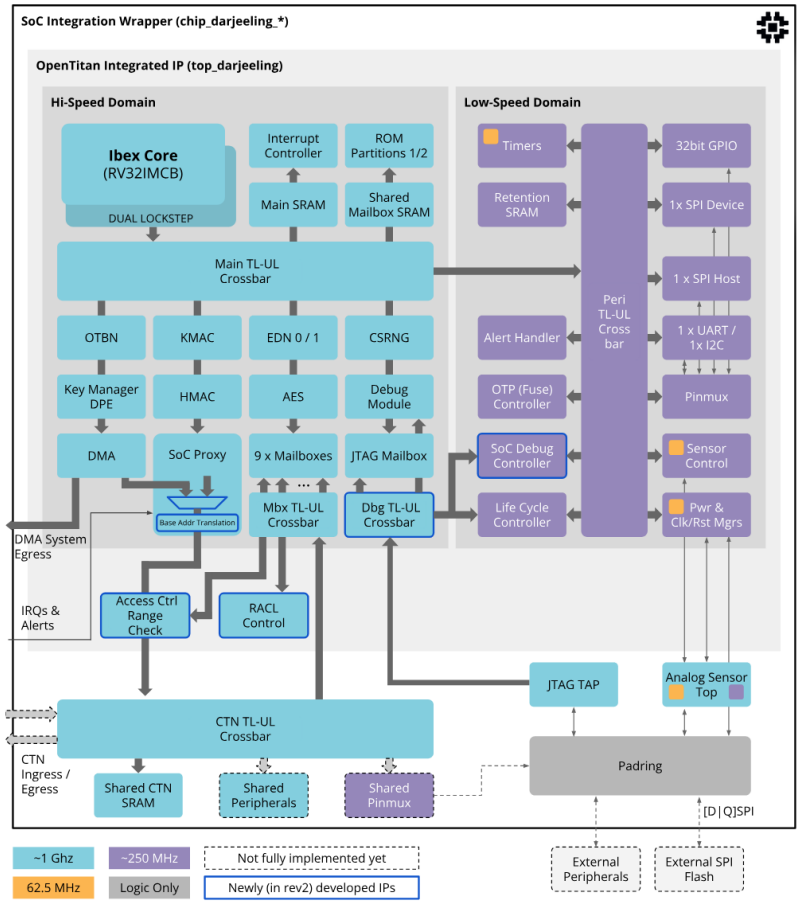


Example Open-Source Projects

OpenTitan

OpenTitan

- "*OpenTitan is an open source silicon Root of Trust (RoT) project. OpenTitan will make the silicon RoT design and implementation more transparent, trustworthy, and secure for enterprises, platform providers, and chip manufacturers*" [https://opentitan.org/book/index.html]
- Repo: https://github.com/lowrisc/opentitan

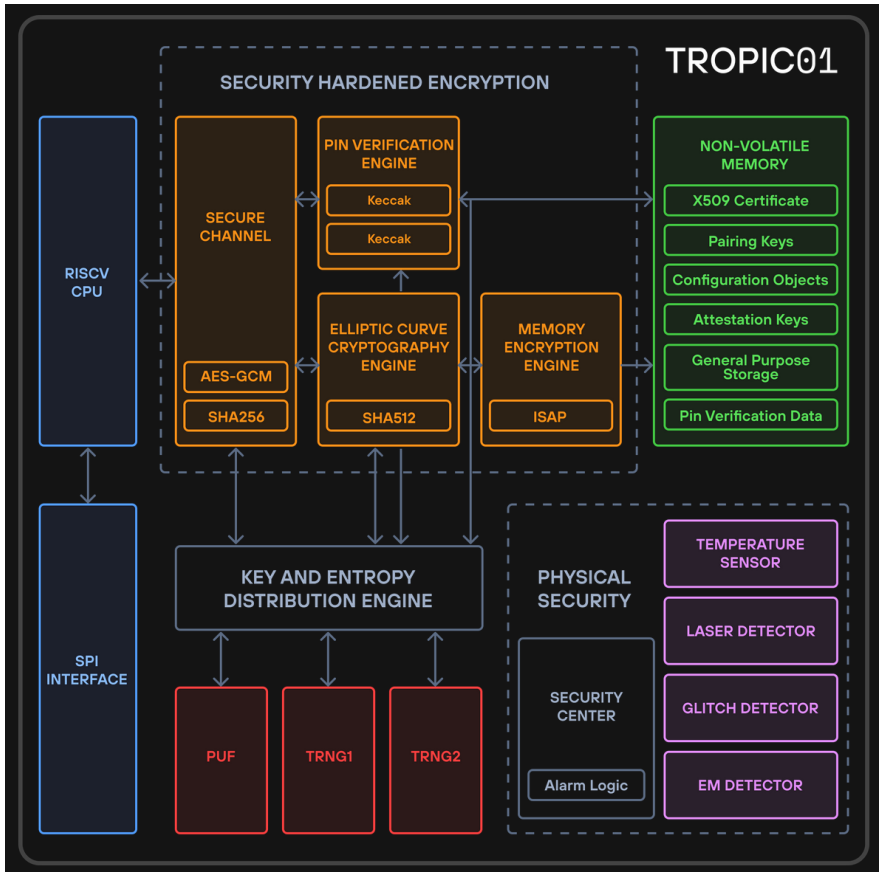


https://opentitan.org/book/hw/top_darjeeling/doc/datasheet.html

TROPIC01

TROPIC01

- *"TROPIC01 is a revolutionary secure element designed to deliver tamper-proof next-gen protection with an open architecture. It sets unprecedented standards for transparency, collaboration, and trust in hardware security."*
- Repo: https://github.com/tropicsquare/tropic01



https://tropicsquare.com/tropic01

Research Areas of Interest

Research Areas

- Post-Quantum Hardware Cryptography Implementations,
  - How to implement algorithms efficiently and securely?
- Chip-to-Chip Secure Communications,
- LLM-based automated formal definition and verification of security properties,
- Memory encryption,
- PUF design and ML-attack resistance,
- RISC-V Security-relate extensions e.g. ISEs, hardware-enforced memory safety, isolation mechanisms etc…
- Integration of ML within security primitives,
  - CFI,
  - Sensor fusion for attack classification.
- Hardware trojan detection methods,
- Secure debug architectures,
- Security within ML accelerators,
- Integration of security features at the (Process Design Kit) PDK level.

Many more….and TUNI is a great place to research these!

https://en.wikipedia.org/wiki/RISC-V

References

- [NT24] Nemiroff, D. & Tokunaga, C. (2024) ‘Fault-Injection Countermeasures, Deployed at Scale’, in *2024 IEEE Physical Assurance and Inspection of Electronics (PAINE)*. [Online]. 12 November 2024 IEEE. pp. 1–7.
- [SZ20] Szefer, J. (2020) *Principles of secure processor architecture design*. [Online]. Cham], Switzerland: Springer.
- [SC17] Schatz, D. et al. (2017) Towards a More Representative Definition of Cyber Security. *The journal of digital forensics, security and law*. [Online] 12 (2), 53–74.
- [LE13] Lee, R. B. (2013) Security basics for computer architects. *Security Basics for Computer Architects*. [Online] 25 (4), 1–111.
- [CO09] Courtois, N. T. (2009) ‘The dark side of security by obscurity: And cloning MiFare classic rail and building passes, anywhere, anytime’, in *SECRYPT 2009 - International Conference on Security and Cryptography, Proceedings*. 2009 pp. 331–338.
- [MS25] Mishra, J. & Sahay, S. K. (2025) Modern Hardware Security: A Review of Attacks and Countermeasures. *arXiv.org*.
- [YI17] Yitbarek, S. F. et al. (2017) ‘Cold Boot Attacks are Still Hot: Security Analysis of Memory Scramblers in Modern Processors’, in *Proceedings - International Symposium on High-Performance Computer Architecture*. [Online]. February 2017 IEEE. p.

References

- [SK05] Skorobogatov, S. (2005) 'Data Remanence in Flash Memory Devices', in Josyula R. Rao & Berk Sunar (eds) *CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS - CHES 2005, PROCEEDINGS*. [Online]. 2005 Berlin, Heidelberg: Springer Berlin Heidelberg. pp. 339–353.
- [KI14] Kim, Y. et al. (2014) 'Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors', in *Proceeding of the 41st annual international symposium on Computer architecuture*. [Online]. 14 June 2014 Piscataway, NJ, USA: IEEE Press. pp. 361–372.
- [KO19] Kocher, P. et al. (2019) 'Spectre Attacks: Exploiting Speculative Execution', in *Proceedings - IEEE Symposium on Security and Privacy*. [Online]. May 2019 IEEE. p.
- [LI20] Lipp, M. et al. (2020) Meltdown: reading kernel memory from user space. Communications of the ACM 63 (6) p.46–56.
- [WS21] Walker, P. & Saxena, N. (2021) 'SoK: assessing the threat potential of vibration-based attacks against live speech using mobile sensors', in *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. [Online]. 28 June 2021 New York, NY, USA: ACM. p.
- [KO11] Kocher, P. et al. (2011) Introduction to differential power analysis. *Journal of cryptographic engineering*. [Online] 1 (1), 5–27.
- [JI24] Jiang, M. et al. (2024) Understanding Vulnerability Inducing Commits of the Linux Kernel. *ACM transactions on software engineering and methodology*. [Online] 33 (7), .

References

- [AS23] Askeland, A. et al. (2023) Who Watches the Watchers: Attacking Glitch Detection Circuits. *IACR transactions on cryptographic hardware and embedded systems*. [Online] 2024 (1), 157–179.
- [YA24] Yang, B. et al. (2004) 'Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard', in *International Test Conference 2004 : proceedings : October 26-October 28, 2004, Charlotte Convention Center, Charlotte, NC, USA*. [Online]. 2004 Piscataway NJ: IEEE. pp. 339–344.
- [TE14] Tehranipoor, M. et al. (2014) *Integrated circuit authentication : hardware trojans and counterfeit detection*. 1st ed. 2014. [Online]. Cham, Switzerland: Springer.
- [SA21] Saarinen, M.-J. O. (2021) 'On Entropy and Bit Patterns of Ring Oscillator Jitter', in *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. [Online]. 1 January 2021 IEEE. pp. 1–6.
- [SZ24] Szymkowiak, T. et al. (2024) 'Poster: Marian: An Open Source RISC-V Processor with Zvk Vector Cryptography Extensions', in *CCS 2024 - Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*. [Online]. 2 December 2024 New York, NY, USA: ACM. pp. 4931–4933.

Questions?