

# RISC-V Cryptography and Hardware Security



**RISC-V**<sup>®</sup>

Technical Sessions - June 29, 2023

**Dr. Markku-Juhani O. Saarinen**  
<mjos@pqshield.com>



# Speaker 🙌

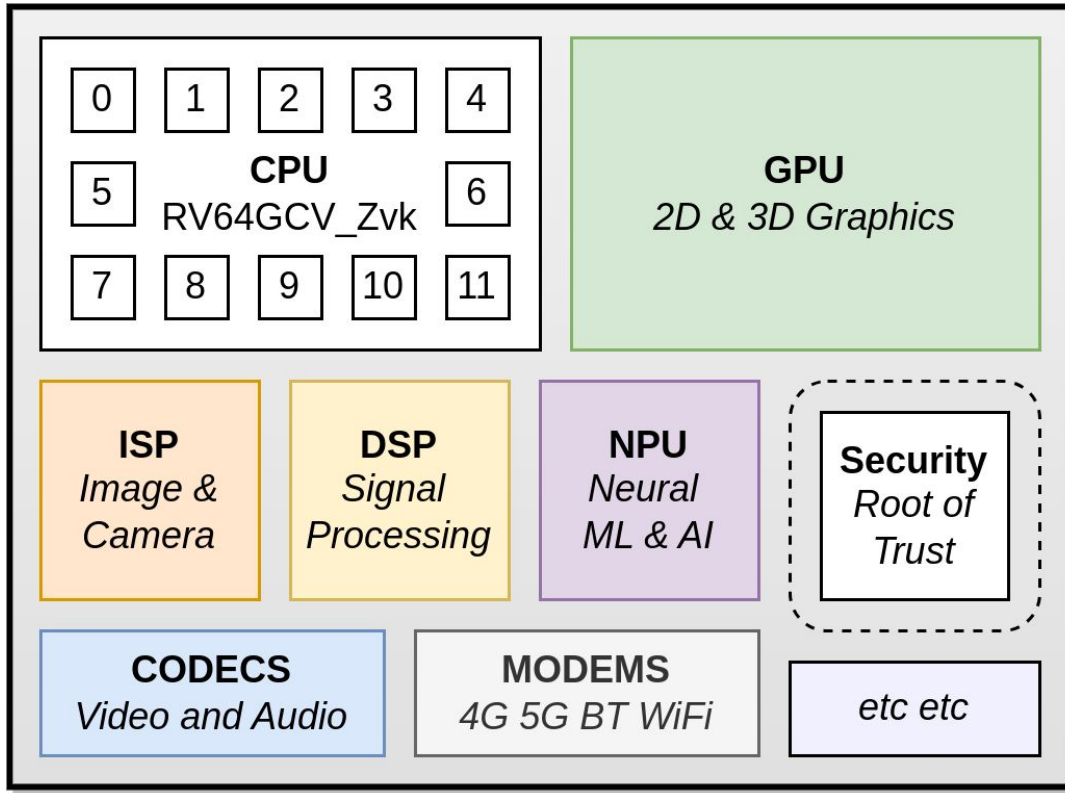
## Markku-Juhani O. Saarinen

- CETG Contributor, Acting Post-Quantum TG Chair
- Staff Crypto Architect (PQShield Ltd, Oxford, UK)
- Professor of Practice (Tampere University, FI)

## *Background: Cryptography & Technical Infosec:*

- Assembler programming since teens (C64 & Amiga!)
- Cryptography as a job for 25+ years (also my PhD)
- Job: PQC HW, Side Channels, RISC-V Crypto

# Big Picture: Cryptography & RISC-V SoC



*Some major hardware components of a mobile device SoC.*

- ISA Extensions in the **main CPU** for applications (e.g. browser), server processes, kernel funcs.
- In an isolated **Root of Trust**. Not ISA. Side-channel secure hardware modules, not directly available to user processes.
- Possibly crypto **Accelerators** in memory IO and storage controllers, modems, or NICs.

**( Fun fact: The GPU, ISP, DSP, NPU, .. , and RoT often also contain RISC-V cores! )**

# ISA: RISC-V Crypto Extensions (“K”)

- **Ratified in Late 2021: Scalar Crypto Extensions**
  - Scalar, resource optimized: AES, SHA2, SM3, SM4, Entropy Source
  - Supporting Bit Manipulations (helps SHA3, Ascon, also CLMUL/GHASH)
  - Data Independent Timing (for scalar)
- **Frozen, in public review until July 23, 2023: Vector Crypto Extensions**
  - Vector, performance optimized: AES, SHA2, SM3, SM4
  - Assorted arithmetic manipulations (+ helps SHA3, Ascon)
  - CLMUL and GHASH, Data Independent Timing for Vector
- **Future:**
  - Full-Rounds AES – for key management / side-channel
  - Post Quantum (Focus on Kyber and Dilithium)
  - Still classical RSA/ECC crypto? Other cipher suites?



**Frozen!**

# Vector Crypto

**In Public Review**

# RISC-V Cryptography Extensions

## Volume II

### ***Vector Instructions***

Version v1.0.0-rc1, 20 June 2023

- Download Vector Crypto frozen specification PDF: <https://github.com/riscv/riscv-crypto/releases>
- Public review runs until July 23. Use the “isa-dev” mailing list or *-even better-* github PRs/issues.

# Scope of The Current ISA Extensions

- AES-GCM is used to process the majority of bulk network traffic in 2023. AES is also common for Storage (disk) encryption – XTS mode.
- Hash function SHA2-256/512 for certificates, HMAC integrity, etc.
- ShangMi in China for similar purposes: SM4 (block cipher), SM3 (hash).
- General bitmanip helps SHA3/SHAKE, Ascon, ChaCha20, other things.

## Advantages:

- Focus on bulk data processing / symmetric crypto. Faster data throughput, better energy efficiency. Security by addressing (timing) attacks.

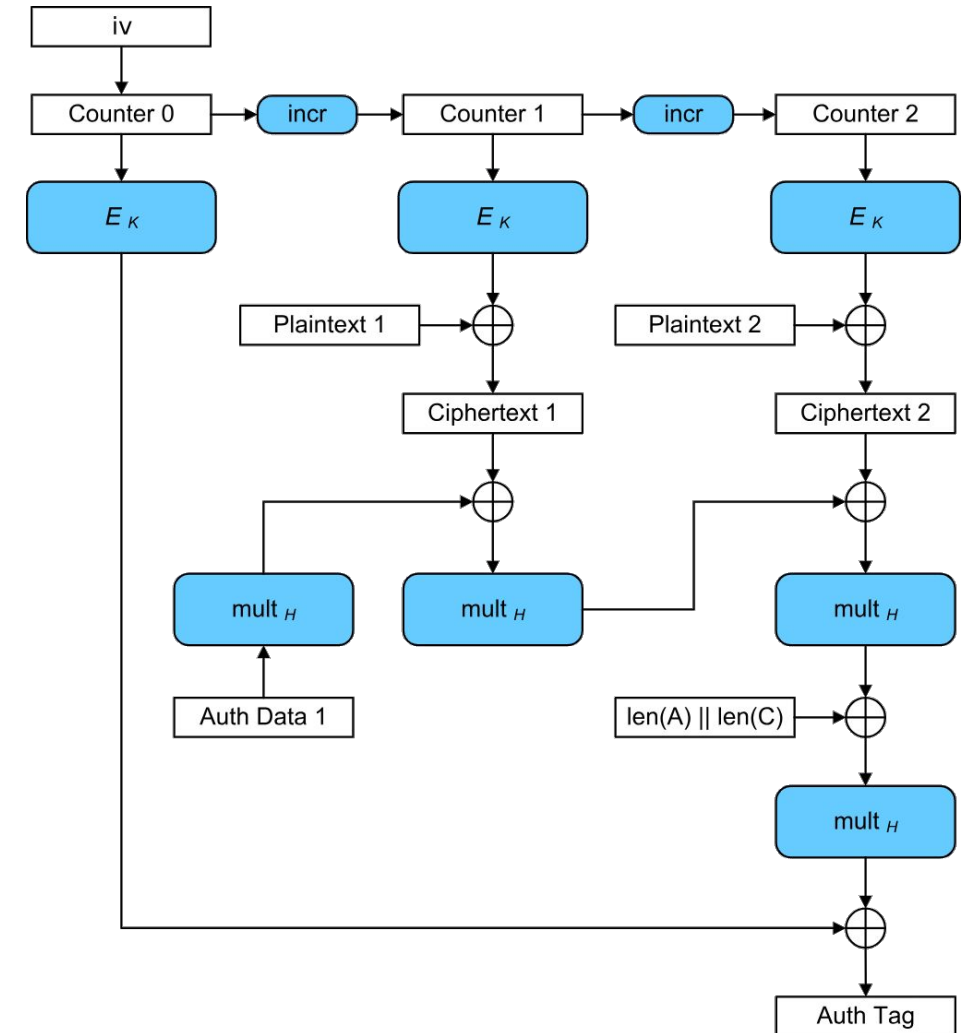
## Not directly addressed:

- Asymmetric Cryptography (key establishment, signatures.)
- Side-channel security beyond timing attacks (DPA, DEMA.)

# AEADs: AES-GCM and SM4-GCM

- AES-GCM: the only “MUST” cipher suite in TLS 1.3 (IETF RFC 8446).
- Confidentiality: AES or SM4 (block cipher  $E_K$ ) in Counter Mode (CTR).
- Data Integrity: Requires 128-bit finite field multiplication ( $\text{mult}_H$ ).

👉 *GCM can be fully parallelized unlike earlier modes like CBC (which only provided data confidentiality.)*





# Vector Extensions refresher

- **32 Architectural Vector registers v0-v31**
  - Each register width (bits): **VLEN** (is  $\geq 128$  for “vk”)
- **Vector register groups**
  - 1, 2, 4 or 8 registers used as a single operand (**LMUL**)
- **Instructions**
  - Memory Load/Store instructions
  - Set configuration (e.g., vl, vtype)
  - Operations on multiple elements
- **Vector Crypto *can* be built on any Vector Extension base**
  - “vk” with  $VLEN \geq 256$  is preferred
  - $ELEN < 64$  or  $XLEN < 64$  block some extensions

# Element Groups (1)

Element Groups provide support for individual data units wider than 64-bits

*New feature, not yet integrated into th main vector spec, but available at:*

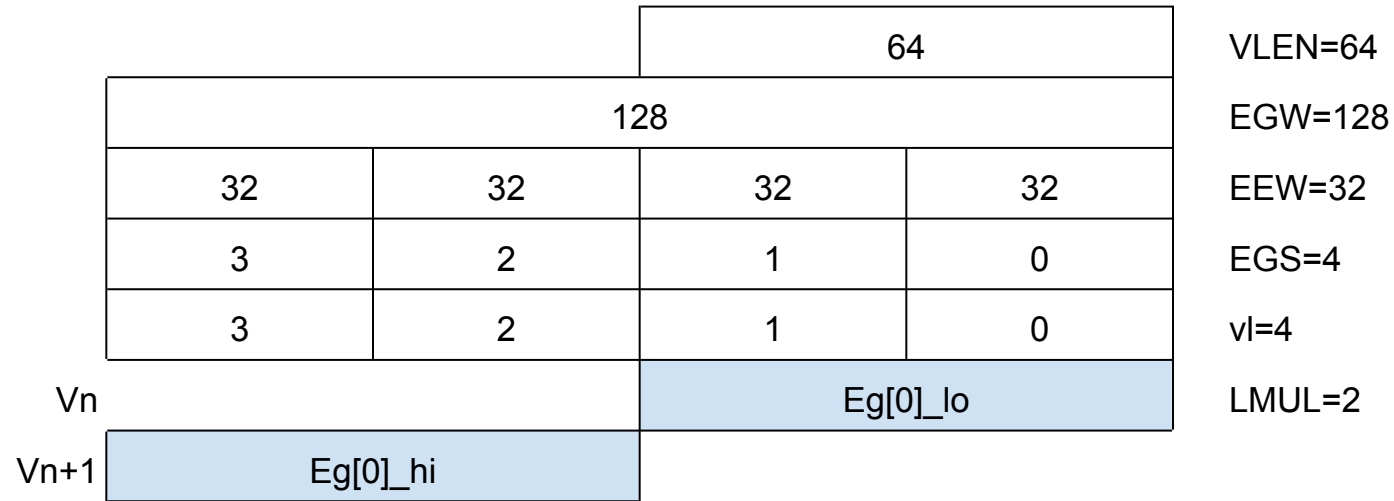
[https://github.com/riscv/riscv-v-spec/blob/master/element\\_groups.adoc](https://github.com/riscv/riscv-v-spec/blob/master/element_groups.adoc)

256								VLEN=256
128				128				EGW=128
32	32	32	32	32	32	32	32	EEW=32
3	2	1	0	3	2	1	0	EGS=4
7	6	5	4	3	2	1	0	vl=8
Eg[1]				Eg[0]				

Vn

- EGW** (Element Group Width): Total number of bits in an element group.
- EEW** (Effective Element Width): Number of bits in each element (=SEW here.)
- EGS** (Element Group Size): Number of elements in an element group.

# Element Groups (2)

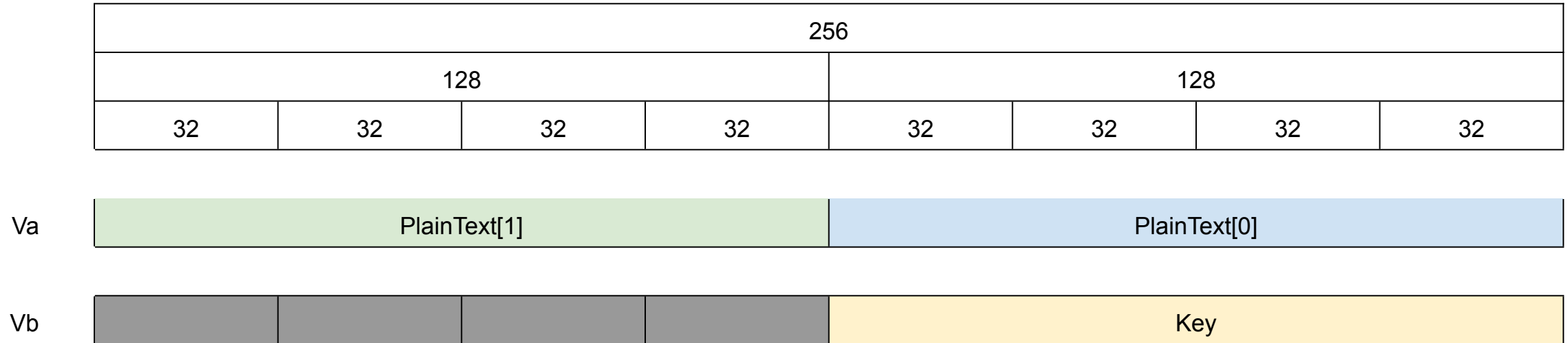


- *Element groups* can cross register boundaries by using *register groups* (LMUL)
  - Enables narrower implementations to support larger EGW instructions
- Elements are still not allowed to cross register boundaries

# Element Groups in Vector Crypto

Instructions	Extension	EGW	EEW	EGS
AES	Zvkned	128	32	4
SHA256	Zvknh[ab]	128	32	4
SHA512	Zvknhb	256	64	4
GCM/GHASH	Zvkg	128	32	4
SM4 cipher	Zvksed	128	32	4
SM3 hash	Zvksh	256	32	8

# Vector-Scalar instructions



- Allows an *element group* to be used as a scalar
- Re-uses the .vs suffix
- Applies scalar to each element group
- > **for example, one AES key could apply to all “lanes”**

# Extension Naming & Profiles

There are ***a lot*** of extensions in the vector crypto specification!  
14 overlapping Zv.. instruction groupings for 27 vector instructions.

Currently to run the PoC tests:

```
$ spike --isa \  
rv64gcv_zvbb_zvbc_zvkg_zvkned_zvknhb_zvksed_zvksh  
pk a.out
```

Clearly not sensible to support  $2^n$  configuration targets for n extensions.  
Will be simplified via “profiles”: <https://github.com/riscv/riscv-profiles/>

# NIST Selective Suites

**Zvkned:** NIST Suite: Vector AES Block Cipher

**Zvknh[ab]:** NIST Suite: Vector SHA-2 Secure Hash

**Zvknc:** NIST Algorithm Suite with carryless multiply

**Zvkng:** NIST Algorithm Suite with GCM

**Zvkn:** NIST Algorithms (Zvkned, Zvknhb, Zvbb, Zvkt)

# ShangMi Selective Suites

- Zvksed:** ShangMi Suite: SM4 Block Cipher
- Zvksh:** ShangMi Suite: SM3 Secure Hash
- Zvksc:** ShangMi Algorithm Suite with carryless mult.
- Zvksg:** ShangMi Algorithm Suite with GCM
- Zvks:** ShangMi Algorithms (Zvksed, Zvksh, Zvbb, Zvkt)



# Common Selective Suites

- Zvbb:** Vector Bit-manipulation used in Cryptography  
*vandn.[vv,vx], vbrev.v, vbrev8.v, vrev8.v, vclz.v, vctz.v, vcpop.v, vro1.[vv,vx], vror.[vv,vx,vi], vwsll.[vv,vx,vi]*
- Zvbc:** Vector Carryless Multiplication  
*vclmul.[vv,vx], vclmulh.[vv,vx]*
- Zvkg:** Vector GCM/GMAC (128-bit fixed modulus)  
*vghsh.vv, vgmul.vv*
- Zvkt:** Vector Data-Independent Execution Latency (DIEL)  
*No extra instructions: Zvkt means that you're asserting DIEL for a set instructions.*

# Zvbb: Vector Bitmanip (drafts: Zvkb)

	<b>Mnemonic</b>	<b>Description</b>
<code>vandn.v[vx]</code>	<code>vs2, [vr]s1, vm</code>	Vector And-Not
<code>vrev.v</code>	<code>vd, vs2, vm</code>	Vector Reverse Bits in Elements
<code>vbrev8.v</code>	<code>vd, vs2, vm</code>	Vector Reverse Bits in Bytes
<code>vrev8.v</code>	<code>vd, vs2, vm</code>	Vector Reverse Bytes
<code>vrol.v[vx]</code>	<code>vd, vs2, [vr]s1, vm</code>	Vector Rotate Left
<code>vror.v[vx]</code> <code>vror.vi</code>	<code>vd, vs2, [vr]s1, vm</code> <code>vd, vs2, uimm, vm</code>	Vector Rotate Right

# More Zvbb: Vector Bitmanip

Mnemonic		Description
<code>vclz.v</code>	<code>vd, vs2, vm</code>	Vector Count Leading Zeros
<code>vctz.v</code>	<code>vd, vs2, vm</code>	Vector Count Trailing Zeros
<code>vcpop.v</code>	<code>vd, vs2, vm</code>	Vector Population Count
<code>vwsll.vv</code>	<code>vd, vs2, vs1, vm</code>	Vector Widening Shift Left Logical
<code>vwsll.vx</code>	<code>vd, vs2, rs1, vm</code>	
<code>vwsll.vi</code>	<code>vd, vs2, uimm, vm</code>	

- `vclz`, `vctz` are useful for arbitrary-precision arithmetic, but not so terribly useful for the sort of “big integer” arithmetic we do in cryptography.
- `vcpop` perhaps more useful in *cryptanalysis* (Kyber CBD sampler only?)

# Zvkg: GHASH for GCM/GMAC

EGW	Mnemonic	Definition
128	<code>vghsh.vv</code> <code>vd, vs2, vs1</code>	Vector GHASH Add-Multiply
128	<code>vgmul.vv</code> <code>vd, vs2</code>	GHASH Multiply

**vgmul** computes  $vd * vs2$  where  $*$  is a 128x128 carryless multiplication reduced to 128 bits it by GHASH's irreducible poly:  $x^{128} + x^7 + x^2 + x + 1$ .

The **vghmac** instruction performs a single iteration of the  $GHASH_H$  algorithm. It computes  $(vd \wedge vs1) * vs2$  with reduction as in `vgmul`.

*(note: vghsh was previously vghmac, had a different order of multiply/add.)*

# Zvbc: Carryless multiply

Mnemonic	Description
<code>vclmul.v[vx] vd, vs2, vs1, vm</code>	Vector Carryless Multiply
<code>vclmulh.v[vx] vd, vs2, vs1, vm</code>	Vector Carryless Multiply Return High Half

- Carryless multiply can be used in arbitrary binary field arithmetic applications, including CRC computation (CoreMark!)
- Main cryptographic use case is an alternative implementation of GCM.
- There are potential applications in code-based & multivariate PQC and some symmetric ciphers, but these are not common in practice.

# Zvkned: AES instructions

EGW	Mnemonic	Description
128	<code>vaesef.v[vs] vd, vs2</code>	Vector AES encrypt final round
128	<code>vaesem.v[vs] vd, vs2</code>	Vector AES encrypt middle round
128	<code>vaesdf.v[vs] vd, vs2</code>	Vector AES decrypt final round
128	<code>vaesdm.v[vs] vd, vs2</code>	Vector AES decrypt middle round
128	<code>vaeskf1.vi vd, vs2, uimm</code>	Vector AES-128 Forward Key Schedule
128	<code>vaeskf2.vi vd, vs2, uimm</code>	Vector AES-256 Forward Key Schedule
128	<code>vaesz.vs vd, vs2</code>	Vector AES round zero (encrypt/decrypt)

- All Vector AES instructions have 2 source operands
- Vd is used as a source to save instruction encoding space

# Zvknh[ab]: SHA-2 instructions

Extension	SEW	EGW	Mnemonic	Description
Zvknha/b	32	128	vsha2ms.vv vd, vs2, vs1	Vector SHA-256 Message Schedule
Zvknha/b	32	128	vsha2ch.vv vd, vs2, vs1	Vector SHA-256 Compression high
Zvknha/b	32	128	vsha2cl.vv vd, vs2, vs1	Vector SHA-256 Compression low

Extension	SEW	EGW	Mnemonic	Description
Zvknhb	64	256	vsha2ms.vv vd, vs2, vs1	Vector SHA-512 Message Schedule
Zvknhb	64	256	vsha2ch.vv vd, vs2, vs1	Vector SHA-512 Compression high
Zvknhb	64	256	vsha2cl.vv vd, vs2, vs1	Vector SHA-512 Compression low

# Zvksed: SM4 Block Cipher

EGW	Mnemonic	Definition
128	vsm4k.vi vd, vs2, uimm	Vector SM4 four Rounds Key Expansion
128	vsm4rv.[vs] vd, vs2	SM4 four Rounds Encryption/Decryption

- vsm4k has 2 source operands (one is an immediate)
- vsm4r has 2 source operands

**SM4** is the Chinese Standard block cipher, with similar external functionality to AES (although only supports a 128-bit key). Internal structure is somewhat different.

Is used for similar purposes as AES: TLS 1.3 with SM4-GCM (RFC 8998), and for storage encryption. Helpful especially for devices in the Chinese domestic market.



# Zvksh: SM3 Secure Hash

EGW	EEW	Mnemonic	Definition
256	32	<code>vsm3me.vv vd, vs2, vs1</code>	Vector SM3 Message Expansion (8 rounds)
256	32	<code>vsm3c.vi vd, vs2, uimm</code>	Vector SM3 Compression (2 rounds)

- `vsm3me` has 3 source operands
- `vsm3c` has 2 source operands
  
- SM3 is the Chinese Hash function standard, with a 256-bit hash (only).
- There is 1 message expansion instruction for every 4 compressions
  - **vslidedown** can be used to provide the current word pair
- This approach was chosen as it is expected to be more performant than having to execute 1 compression instruction per word pair.

# Try it out! Testing Vector Crypto

- There's PoC code and tests for AES, SHA, SM3/SM4 in the riscv-crypto repo: [doc/vector/code-samples](#)
- For experiments: You can use at least mainline Spike, LLVM 17. Not yet prepackaged – some time/effort needed. There's also QEMU and GCC support.
- OpenSSL PR exists (parts being pulled into Kernel?)  
<https://github.com/openssl/openssl/pull/20149>

# What about Scalar Crypto?

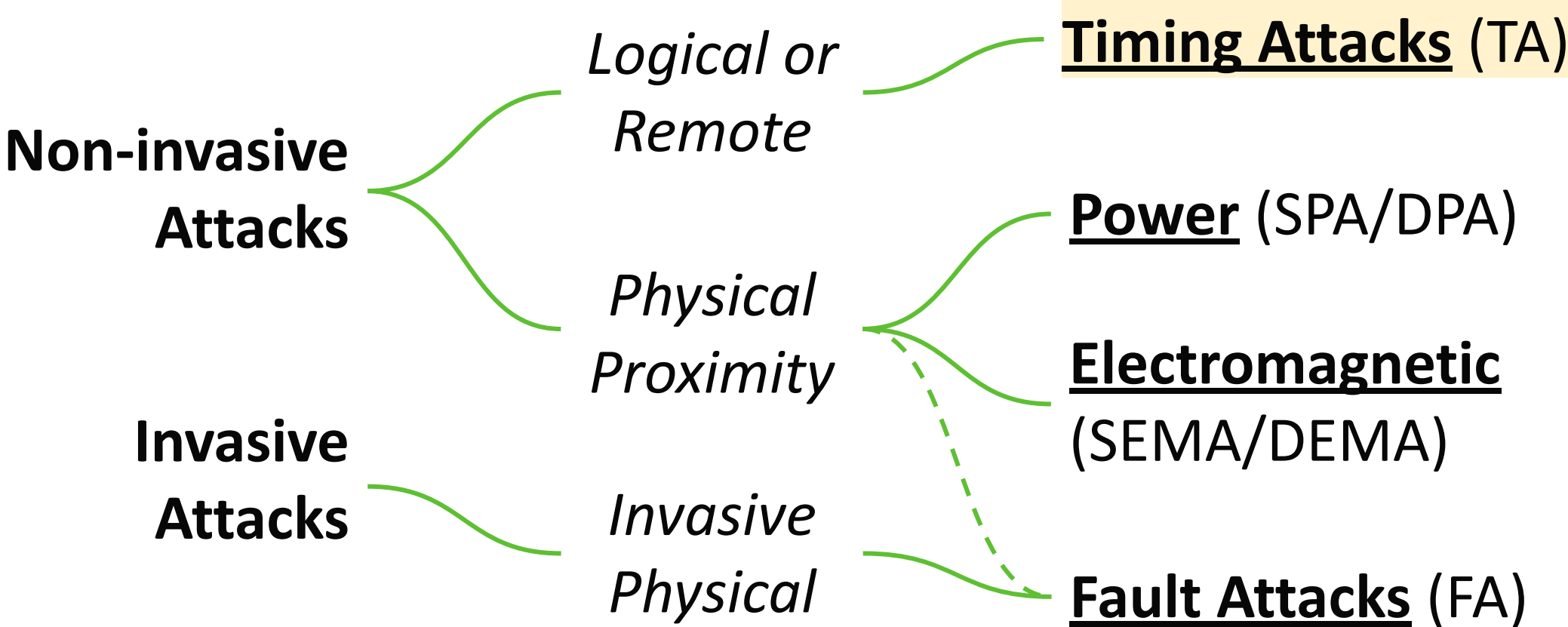
- Same basic features, but uses general purpose registers. Can be added to a RV32 or RV64 processor without vector support.
- The hardware area footprint / energy consumption of scalar cryptography is really minimal compared to vector crypto.
- Building a *constant time* AES or GCM without hardware support is very difficult (slow) – looking at 10x perf loss.
- The **Zkr Entropy CSR** is in the scalar crypto spec, but applies equally for building (D)RBGs using vector crypto instructions.

# Zkt & Zvkt DIEI

“Constant Time”

# Side-Channel Attacks

## Some Security Requirements



# 25+ Years of Timing Attacks

## Examples over the years:

- P.C. Kocher: *"Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems."* (CRYPTO 1996. Target: RSAREF 2.0 running on MS-DOS.)
- D. Brumley and D. Boneh: *"Remote timing attacks are practical."* (USENIX Security 2003. OpenSSL RSA remote key recovery, CVE-2003-0147.)
- B. Brumley and N. Toveri: *"Remote Timing Attacks Are Still Practical."* (ESORICS 2011. OpenSSL ECDSA remote key recovery, CVE-2011-1945.)
- *.. mature crypto implementations (e.g. OpenSSL) are nowadays mostly okay.*

## But new stuff keeps happening:

- Q. Guo, T. Johansson, A. Nilsson, *"A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on FrodoKEM."* (Crypto 2020.)

*Addressed via constant-time implementation techniques.*

# Sources of Timing leaks

1. Secret-controlled branches and loops:

```
if <secret> then { delay1(); } else { delay2(); }
```

2. Memory accesses (cache timing attacks). Can be a load or store.

```
ct = SBox[pt ^ key]; // observe latency with different inputs.
```

3. Arithmetic operations whose processing time just depends on inputs

```
x = y % q; // division and remainder ops are rarely constant-time.
```

*Need Data Independent Execution Latency to process data.*

# Data Independent Execution Latency

- RISC-V CETG codified data independent timing as the Zkt extension for scalar, and the (*new*) Zvkt DIEEL list for vector (**this spec**).
- Vendor asserts Zkt/Zvkt: Contract between programmer and the device includes DIEEL for specific instructions.
- One can use static analysis or dynamic variable tainting (in emulator) to verify that compiled code is using only the right (DIEEL) instructions to handle secret data.
- Contrast with “*Constant-timeliness*” of Intel and ARM instructions: until recently derived from experiments. + A lot of platforms..



# Zvkt DIEL Listings in Section 2.14

1. All dedicated crypto instructions (Zk\*) are DIEL.
2. Vector Bitmanip in Crypto Spec: Zvbb, Zvbc
3. General arithmetic: Add/sub, compare and set, copy, extend, Boolean, multiply, multiply-add, integer merge, shift
4. With limitations (only “data” registers): permute, slide

## Excluded:

- All Load/store, floating point operations
- Clip, compress, divide, remainder, average, mask op, min/max, multiply-saturate, reduce, shift round, vset, ..

# RISC-V Summit

June 7, 2023

## Android on RISC-V Progress & Updates

Lars Bergstrom, PhD  
Director of Engineering, Android



android

# RISC-V Android ABI Progress and Wishlist

See our current progress here: <https://github.com/google/android-riscv64>

Known Issues here: <https://github.com/google/android-riscv64/issues>

Join the Android SIG mailing list and come to the monthly meetings for more: <https://lists.riscv.org/g/sig-android>

What's next after “rva22 + vector + vector crypto”?

First: need to make sure to land vector crypto!

- Still haven't voted on ratification at time of writing (<https://github.com/riscv/riscv-crypto/releases>)


Very excited for platform support for the following extensions, but unclear if it's *required* for Android applications as well...

- Zjid instruction/data consistency for JIT
- Zisslpcfi for security
- Zjpm pointer masking for hwasan
- Hans Boehm's proposed new atomics
- Bfloat16 vector support

# Upcoming ISA

**CETG Plans..**

# Work Item 1: All-Rounds Vector AES

- **All rounds** of AES-128/256 with a single instruction – allows more robust implementations than per-round.
- Manage keys with opaque identifiers (“handles”).  
 *Helps against side-channel and cold-boot attacks.*
- Methods for key import/export/wrapping are still under discussion. Masked keys? CSR Interface? Key mgmt ISA?
- Roughly similar in purpose to Intel Key Locker (“AES KL”.)

# Work Item 2: Post-Quantum

- RSA and Elliptic Curve are decimated by quantum (Shor's).
- Post-Quantum Cryptography (PQC) = Cryptography that is not vulnerable to a quantum algorithms.
- Symmetric cryptography is not affected much (Grover's) – most current RISC-V Zk extensions are actually fine.
- ~2015 U.S. Government decided to transition to PQC.  
NIST Standardization of PQC 2016-2024.

# Recap: (July 2022) NIST Selects PQC Algorithms

## Post-Quantum Crypto transition affects all Crypto Applications

**NIST Post-Quantum Crypto: Selected July 2022, Standards 2024.**

### **Kyber (+ Round 4 KEMs)**

Replaces EC(DH), RSA key establishment.

### **Dilithium, Falcon, SPHINCS+**

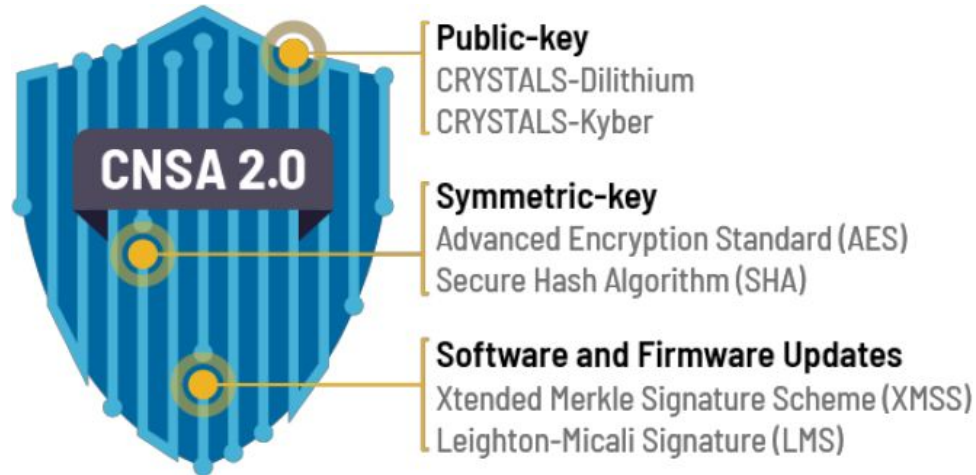
Replaces EC(DSA), RSA signatures.



### ***Especially for U.S. Government Entities:***

- *Active transition effort expected (presidential directives NSM-08, NSM-10).*
- *Regulations mandate FIPS 140-3 cryptography -> also for PQC modules.*

# PQC in National Security Systems



## Transition 2025-2030-2035:

*“Note that this will effectively deprecate [in NSS] the use of RSA, Diffie-Hellman (DH), and elliptic curve cryptography (ECDH and ECDSA) when mandated.”*

*Table III: CNSA 2.0 quantum-resistant public-key algorithms*

Algorithm	Function	Specification	Parameters
CRYSTALS-Kyber	Asymmetric algorithm for key establishment	TBD	Use Level V parameters for all classification levels.
CRYSTALS-Dilithium	Asymmetric algorithm for digital signatures	TBD	Use Level V parameters for all classification levels.



# Kyber & Dilithium: Arithmetic

- NTT (Number Theoretic Transform) uses butterfly operations and a mul/add/sub mod fixed special  $q$ :  
 $q=0xD01$  (Kyber)  $q=0x7FE001$  (Dilithium)
- Better SHAKE/SHA3: On an ARM microcontroller ~50% cycles is spent on the Keccak Permutation.
- Rejection sampling / bit gather (esp. for A matrix).
- Bitmanip (16- and 32-bit) shifts for CBD, bit packing.

# Vector NTT: (mod q) arithmetic

- Typical Montgomery technique used for (mod q) multiplication requires widening, extra reduction multiplication, shift, add.
- Proposal: “*Vector Single-Width mod q Integer Multiply-Add Instructions*” (vmaccq, vnmsacq, vmaddq, vnmsubq).
- Reduces instruction count for NTT significantly, avoidance of widening also reduces register pressure.

**Example:** vmaccq.vv vd, vs1, vs2, vm

#  $vd[i] = +(vs1[i] * vs2[i]) + vd[i] \pmod{q}$

If SEW=16:  $q=0xD01$ , else if SEW=32:  $q=0x7FE001$ .

(Or perhaps select modulus q in a special CSR register.)

# Better SHA3 / Keccak? Samplers?

Scalar and vector crypto already have some SHA3 support.

- `vandn` (Chi), `vrol` (Theta) are in Zvk for Keccak.
- with `vrgather[ei]` (for Pi) this is reasonably good.
- Do we need even more speedup for permutation?
- SHA3 is also needed for SPHINCS+, XMSS, LMS/HSS.

Look at gather / compress sequences for samplers:

- Dilithium: Extract a 24-bit segment, clear high bit (bit 23), compare and select if  $x < q$ , expand to 32 bits for use.
- Kyber: 12-bit segment  $x$ , select  $x$  if  $x < q$ , expand to 16 bits.

# PQC Extensions Workplan Q3/23-Q2/24

1. Elect officials (I'm the Acting Chair, Richard acting VC)
2. Initial focus on Kyber and Dilithium which are a transition priority. NIST Should be releasing Draft PQC Standards in mid-2023 (*yes, very soon*) for Kyber and Dilithium.
3. Do quantitative analysis with real-life benchmarks from PQ TLS ciphersuites, certificate processing, etc use cases
4. Don't propose instructions unless they show advantage
5. Proceed into freeze by the time NIST PQC Standards review is complete (not many changes expected)
6. Try to time RVI ratification shortly after NIST's ratification

# Non-ISA: Physical / Platform Security

Root of Trust, etc

# Physical & Platform Security: Examples

1. We expect mobile devices to retain security (user privacy, authentication credentials) even if physically lost / stolen.
2. Makers of IoT devices and Appliances want to protect them against content piracy, unauthorized modifications, etc.
3. Cloud computing customers hope for assurances of data confidentiality, even if the host platform is attacked.
4. Detecting trojans and backdoors inserted into unsupervised devices (or inserted into/via outsourced or OEM components.)

# Security Model TG (*meeting right now..*)

- RISC-V Security Model: Worked on by the namesake Task Group.
- Defines common terminology (and possibly concrete technical requirements) for security components and Root of Trust (RoT).
- Work in the TG is ongoing but I expect the model to be compatible with industry/market best practices.

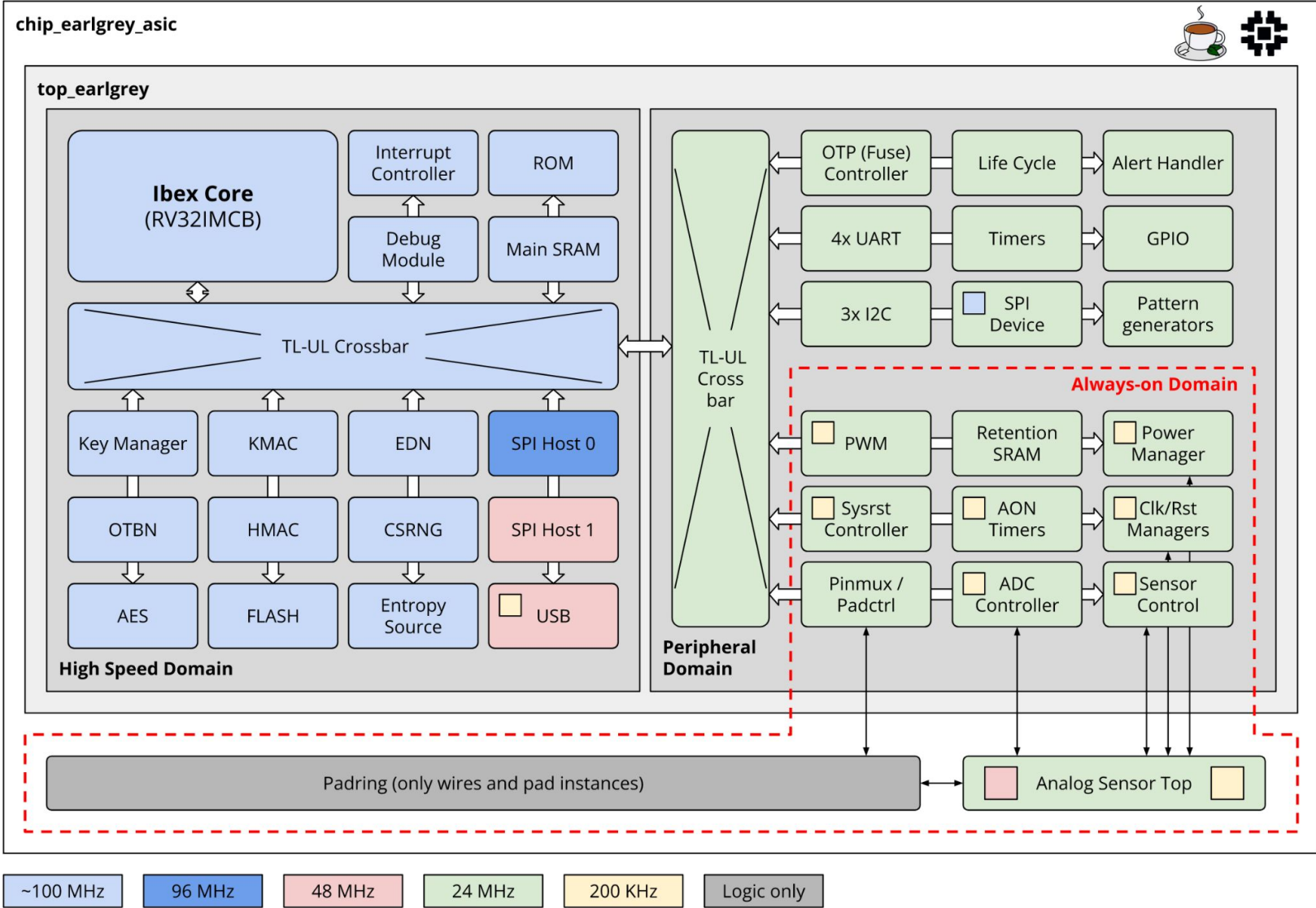
<i>Unique Identification of devices</i>	<i>Prevent Update Rollback</i>
<i>Security Lifecycle</i>	<i>Isolation (trusted vs untrusted)</i>
<i>Attestation / Trustworthiness</i>	<i>Appropriate Isolation Interfaces</i>
<i>Authorized Code Execution</i>	<i>Data/credential binding to device</i>
<i>Secure Update / Supply Chain</i>	<i>Crypto services / Minimalism</i>

# OpenTitan: A Root of Trust Project

- **OpenTitan** is an open source Root of Trust (RoT) project. Three defined discrete IC use cases are given:
  - Platform Integrity Module (e.g. secure boot)
  - Trusted Platform Module (TPM 2.0)
  - Universal 2nd Factor Security Key (USB U2F FIDO)
- However the design is very modular, like a “toolbox” almost.
- Has a RISC-V Core (IBEX), all kinds of crypto peripherals (self-developed). But not for offloading crypto ops from host!
- Open Source: <https://github.com/lowRISC/opentitan>



# OpenTitan (Discrete Chip Configuration)

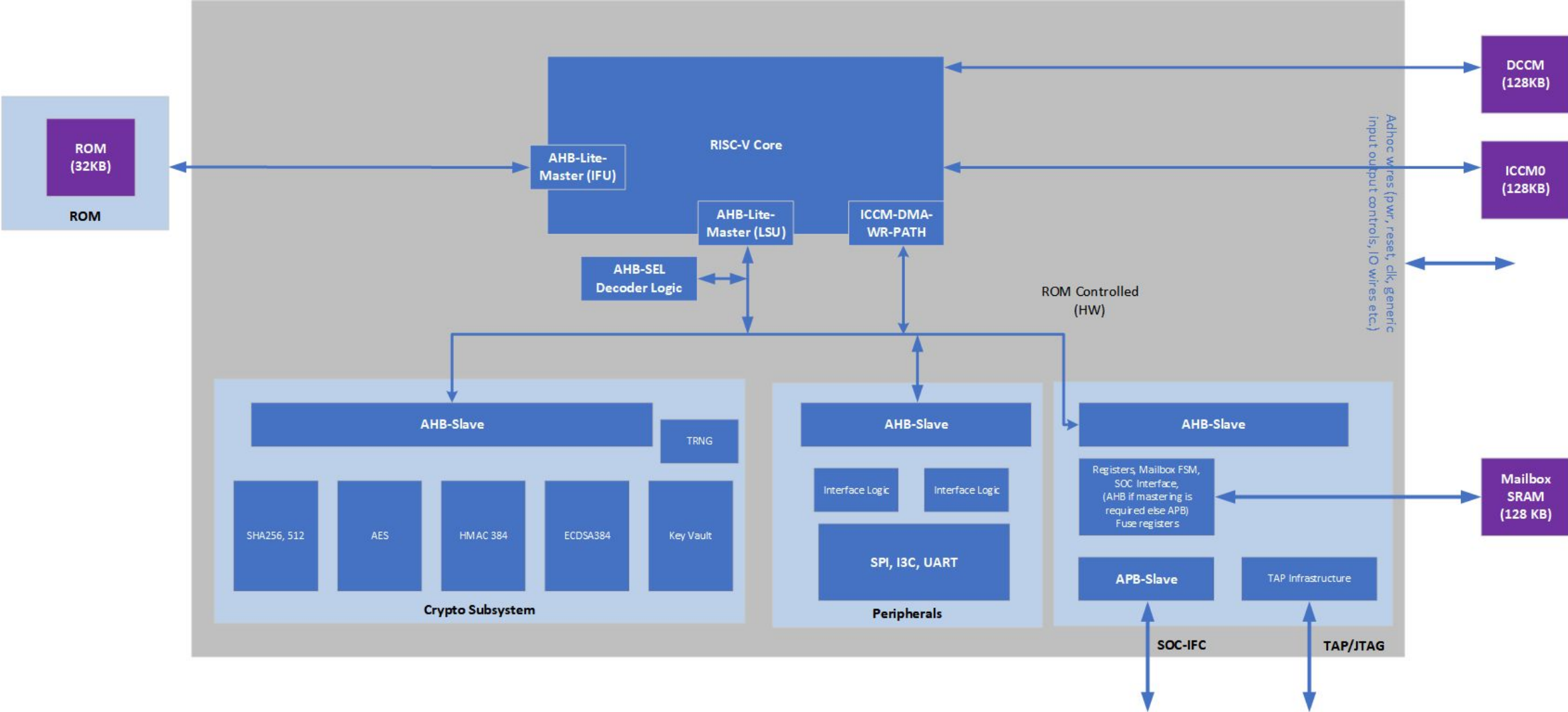


# Caliptra: Securing the SoC


- **Caliptra** is primarily a Root of Trust (RoT) for Measurement.  
*“Targets datacenter-class SoCs: CPUs, GPUs, DPUs, TPUs.”*
- Secure Boot (“measure” code and configuration). Also offers some (identity) services to the host SoC.
- Has a RISC-V Core (VeeR), crypto peripherals (some from OpenTitan.) Again, not for offloading crypto ops from CPU.

Open Source: <https://github.com/chipsalliance/Caliptra/>

# Caliptra Core Block Diagram



# On RoT Security Requirements

- Root of Trust (RoT) generally requires advanced Side Channel Attack (SCA) and Fault Attack / Fault Injection resistance.
  - Traditionally (userspace) software cryptography has much looser invasive/non-invasive physical security requirements.
  - Even a “good” random number generator is pretty useless if not FIPS/CC certifiable (SP 800-90A/B/C and AIS-20/31.)
-  *Standard (FIPS 140-3, Common Criteria) requirements determine many of of RoT low-level design features.*

# Side Channels: What needs to be protected?

## FIPS says CSPs – “Critical Security Parameters”

### Classification in Crypto Module world:

- Public Security Parameter (**PSP**) needs integrity only: Can't be modified.
- Critical Security Parameter (**CSP**) needs confidentiality (secrecy) and integrity.
- Together these are Sensitive Security Parameters (**SSP**  $\approx$  *All variables in crypto!*)

**Section 7.8 of ISO/IEC 19790:2012(E) and 19790:2022(E):** *“Non-invasive attacks attempt to compromise a cryptographic module by acquiring knowledge of the module’s **CSPs** without physically modifying or invading the module.”*

**For us, a **CSP** is any information that helps (the attacker) directly or indirectly to:**

1. Determine a shared secret in a key establishment scheme or
2. Forge a signature in a signature scheme.

# How do we test SCA-secure hardware in design?

## Physical testing is just the final step

- 1. Design secure gadgets.** Arithmetic operation “gadgets” should be *provably secure* in appropriate model (t-probing model, noisy leakage model), perhaps have SNI (Strong Non-Interference) composability, etc.
- 2. Leakage simulation** in microcontroller and **Pre-Silicon** for hardware.
  - The models range from very simple & fast – based on bit toggling – to extremely advanced “3D” physical models.
  - Attack modeling with leakage simulation can be very advanced.
- 3. Physical verification / Sign-Off.** Running *leakage assessment* tests like TVLA. Mainly to find implementation-specific effects.

# Certification: FIPS 140-3 vs. Common Criteria

## Standardized Checks vs. Penetration Testing

- **FIPS 140-3** (“Security Requirements for Cryptographic Modules”) Mostly a checklist / functional testing approach. Levels 3 and 4 mandate “*non-invasive attack mitigation*” testing “*if claimed.*”
- **Common Criteria (CC)** can mean many things! High-assurance **Protection Profiles (PP)** contain **AVA\_VAN.4** or **.5** (Advanced) methodical vulnerability analysis with “*attack potential*” scores.
- **NSS (U.S. DoD / IC) NIAP** also defines Common Criteria Protection Profiles, but borrows many things from FIPS testing.

# Common Criteria: AVA\_VAN.5

## Evaluation of “High Attack Potential”

- **Score-based system.** High attack potential (well-resourced) lab spends 1-3 months, assigns an score ( $\approx$ \$ cost) on attack: *required time, expertise, knowledge/access of TOE, equipment, samples.*
- Covers things like (machine learning) template attacks, but is agnostic to PQC vs Classical !
- Practical: Aims at **key recovery** or similar break.

Used with Smart Cards and similar devices:

<https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3.2.pdf>

Tool	Equipment
Low-end light injection (UV, flash light)	Standard
Electrical glitches workstation	Standard
Binocular microscope	Standard
Thermal stress tools	Standard
Voltage supply	Standard
PC or workstation	Standard
Software tools (fuzzing, test suite)	Standard
Code static analysis tools	Standard
Low-end oscilloscope	Standard
High-end GPU card	Standard
Signal analysis tools	Standard
EMFI, FBBI workstations	Specialized
Optical microscope	Specialized
3D X-Rays workstation	Specialized
Micro-probing workstation	Specialized
High-end laser workstation	Specialized
Real time pattern recognition system	Specialized
High-end oscilloscope	Specialized
Spectrum analyser	Specialized
Wet chemistry tooling (acids & solvents)	Specialized
Dry chemistry (Plasma)	Specialized
Micro-milling and thinning machine	Specialized
Low-end Scanning Electron microscope (SEM)	Specialized
EM signal acquisition workstation	Specialized
Low-end Emission Microscope (EMMI)	Specialized
Low-end Focus Ion Beam (FIB)	Specialized
High-end Scanning Electron Microscope (SEM)	Bespoke
Atomic Force Microscope (AFM)	Bespoke
High-end Focused Ion Beam (FIB)	Bespoke
New Tech Design Verification and Failure Analysis Tools	Bespoke
High-end Emission Microscope (EMMI)	Bespoke
Chip reverse engineering workstation	Bespoke

Table 10: Categorisation of Tools



# SP 800-140Fr1 & New ISO 19790 → ISO 17825

UNCLASSIFIED / NON CLASSIFIÉ

ISO/IEC WD 19790:2022(E)

## Annex F (normative)

### Approved non-invasive attack mitigation test metrics

#### Purpose

This Annex provides a list of the ISO/IEC approved non-invasive attack mitigation test metrics applicable to this document. This list is not exhaustive.

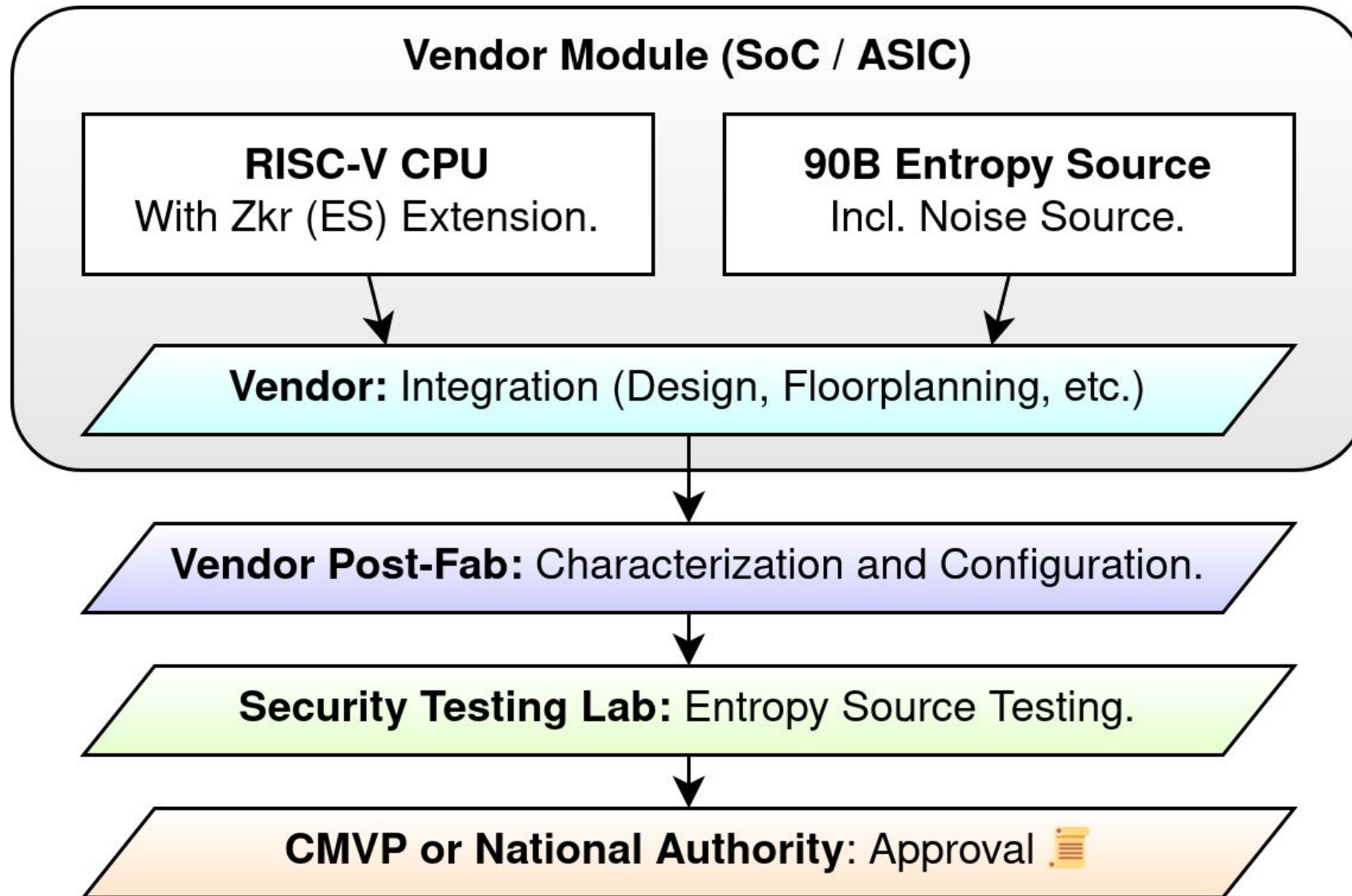
This does not preclude the use of approval authority approved non-invasive attack mitigation test metrics.

An approval authority may supersede this Annex in its entirety with its own list of approved non-invasive attack mitigation test metrics.

#### F.1.1 Non-invasive attack mitigation test metrics

- a) ISO/IEC 17825 *Information technology – Security techniques – Testing methods for the mitigation of non-invasive attack classes against cryptographic modules.*

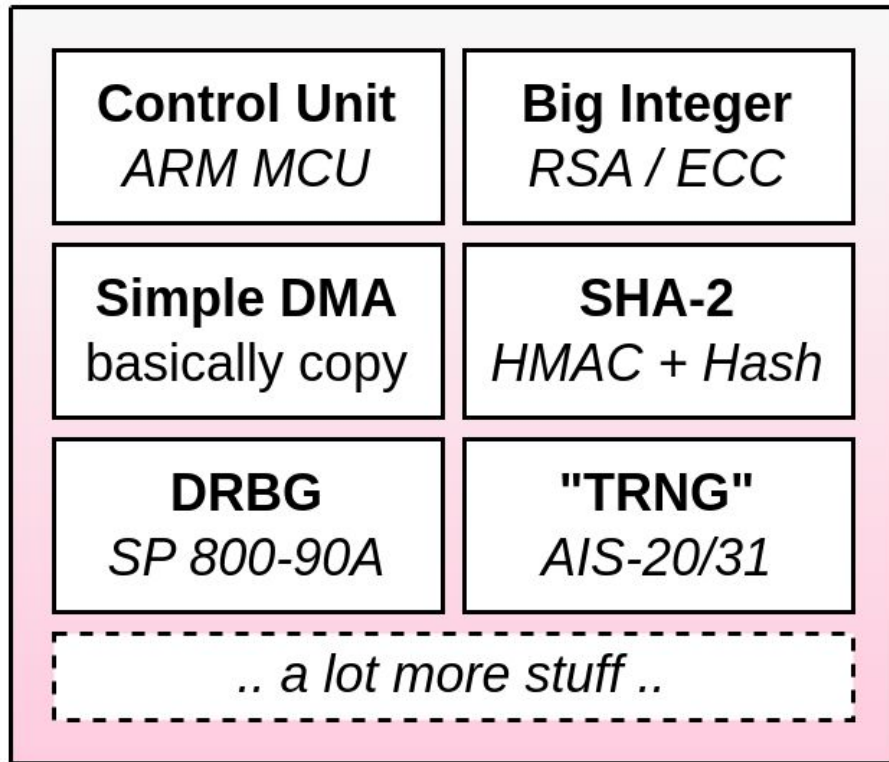
# RISC-V + Entropy Source Integration



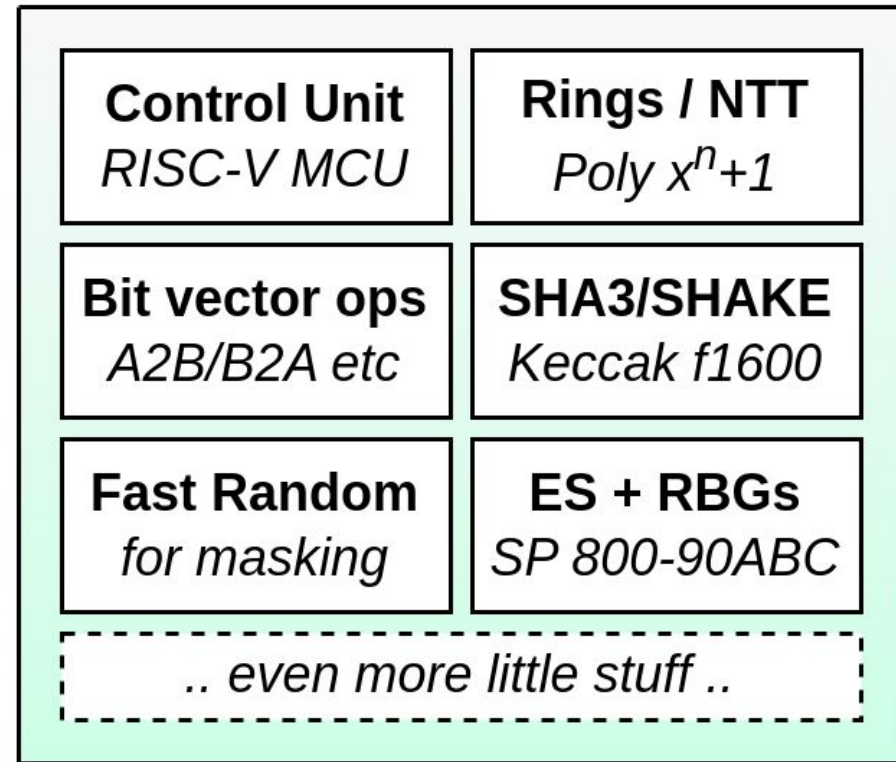
# PQC: Secure Elements Evolve Too

RoT core cryptography may need an update after 2023 ..

Generic Secure Element in -2020



Generic Secure Element in 2025-



# Thanks!

Questions?