Building and Testing a Modern TRNG/RBG: The RISC-V Entropy Source Interface

Think openly, build securely

ICMC 2021

September 3, 2021

Dr. Markku-Juhani O. Saarinen

Senior Cryptography Engineer, PQShield Ltd.

Outline: The RISC-V Entropy Source

1. **Cryptography/Driver Developer: The RISC-V ISA Specs.**
2. Chip Vendor: On Integration of ES Modules & RISC-V CPUs.
3. Security testing lab: Raw sample interfaces (“GetNoise”).

RISC-V ISA Specs

Terminology

- **RISC-V** is an **Instruction Set Architecture** (ISA) in the same sense as ARMv7, ARMv8, i386, or x86 are. Have Compiler & Linux support etc.

- **Open standard**: Many public and commercial designs for RISC-V CPU chips. Many vendors and variants: 32-bit, 64-bit, Vector, FPU, etc.

- Anyone can add *Custom ("X") Extensions*, but for interoperability the **RISC-V International (RVI)** maintains a set of *standard extensions*.

- **Scalar Cryptography & Entropy Source** is a new standard **"K"** extension from the CETG (Cryptography Extensions Task Group).

Get the Spec! It's *free to use.*

September 2021: Frozen

RISC-V Cryptography Extensions Volume I

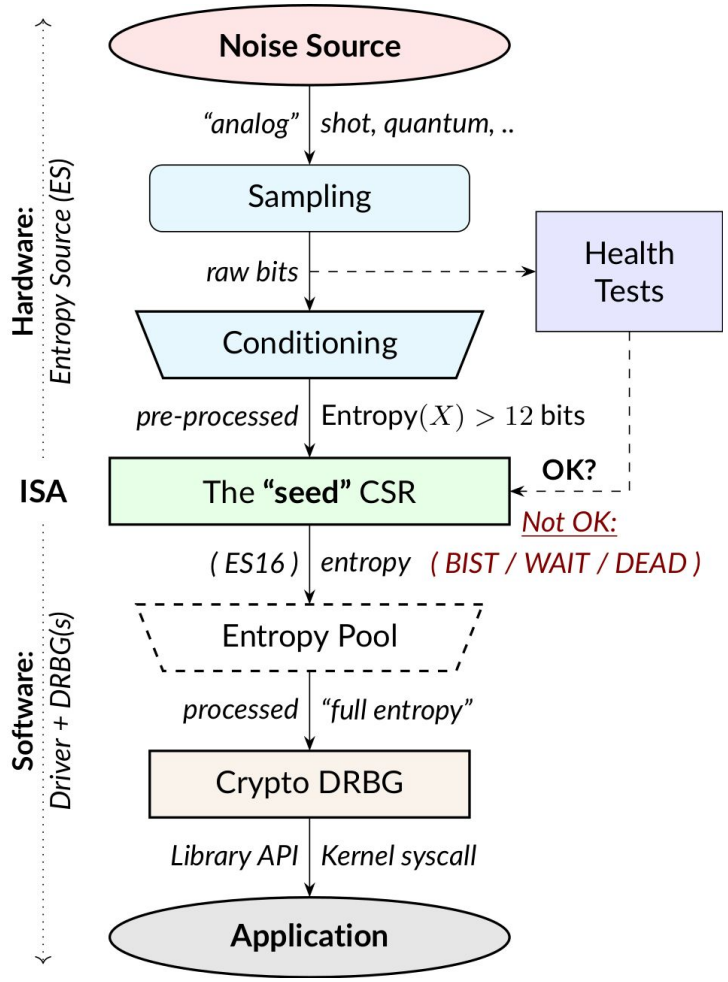
Scalar & Entropy Source Instructions

https://github.com/riscv/riscv-crypto/releases

RISC-V RBG Architecture in Short

The “seed” Control and Status Register (CSR)

- **Not** a PRNG/DRBG like Intel RDRAND!
- An on-chip Physical Entropy Source with >0.75 min-entropy (>0.997 Shannon).
- Requires (2:1) cryptographic conditioning (on driver/sw side) for 90C *“Full Entropy.”*
- Can be used to **seed** a FIPS DRBG / RBG2, RBG3, or by (Linux) Kernel random.
- Both Linux and Microcontroller class.



Accessing the RISC-V Entropy Source

The “seed” Control and Status Register (CSR)

Accessed via control and status register **seed** (CSR 0x015). Assembler:

```
csrrw rd, seed, x0
```

- Instruction returns a word in **rd**, with a 2-bit status + 16 bit entropy.
- Access type must be a *read-write* (for flushing the entropy value).
- CSR is is access controlled with mseccfg.**sseed** and mseccfg.**useed** bits. *M mode may be the only mode in a Microcontroller implementation.*

Polling Randomness from “seed” CSR

High bits **seed[31:30] = OPST** indicate status:

00	**BIST**	Built-In Self-Test.
01	**WAIT**	Sufficient entropy is not yet available.
10	**ES16**	*Success*: Have 16 bits in seed[15:0].
11	**DEAD**	An unrecoverable self-test error.

CSR accesses return immediately, so the interface needs to be repeatedly *polled* (this can be opportunistic -- e.g. in interrupts).

ES16: Return bits always meet specific min-entropy requirements.

Accessing the RISC-V Entropy Source

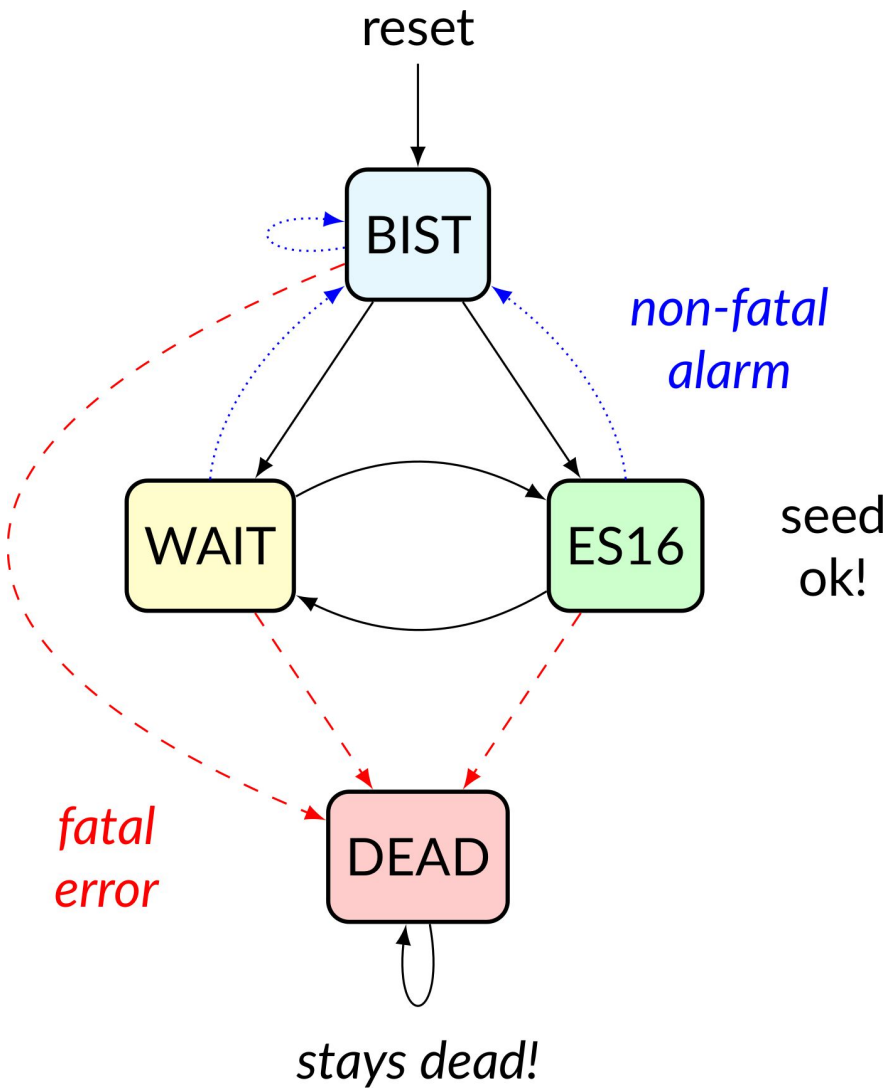
The “seed” Control and Status Register (CSR 0x015)

Bits	Name	Description
31:30	OPST	Status: BIST (00), WAIT (01), ES16 (10), DEAD (11).
29:24	*reserved*	For future use by the RISC-V specification.
23:16	*custom*	Designated for custom and experimental use.
15: 0	entropy	16 bits of randomness, only when OPST=ES16.

Examples:

0x00000000	**BIST** status, no entropy.
0x8000ABCD	**ES16** status, entropy = 0xABCD.
0xFFFFFFFF	**DEAD** status, no entropy.

Polling Randomness from “seed”



- After startup (**BIST**), the state is typically either **WAIT** or **ES16**.
- Ignore **WAIT**, concatenate **ES16**.
- When sufficient amount of entropy is gathered: *Condition (hash) it, and use it to (re)seed a* ***DRBG*** *(SP 800-90A Deterministic Random Bit Generator).*

Implementation Requirements

For successfully polled ES16 “seed” bits

Physical implementations should be of (1) 90B or (2) PTG.2 types.

1. **SP 800-90B IID or non-IID Entropy Source**: 192 bits of assessed min-entropy per 16*16 = 256-bit output block (rate 0.75). With 2:1 driver conditioning this will yield 128 bits of “full entropy”.

2. **BSI AIS-31 PTG.2 Source.** In addition to 90B requirements, also PTG.2 requirements, including 0.997 bits of Shannon entropy.

3. **Virtual machines.** We want emulation to have security too. Host source must have at least PQC Category 5 / 256-bit security level.

Access Control to Physical Sources

Some Security Considerations on Direct Access

Typically this interface is not available in user mode. Supervisor mode Linux kernel may be direct (or trap & emulate if not entirely trusted).

- **Depletion.** Active polling can deny entropy source to all other consumers. Trap & emulate with a DRBG -> harder to deplete.
- **Covert Channels.** Direct physical access may be used to establish communication channels across security boundaries.
- **Hardware Fingerprinting.** Entropy source (and its noise source circuits) may have a uniquely identifiable hardware “signature”.

Implementation Requirements

Health Tests etc

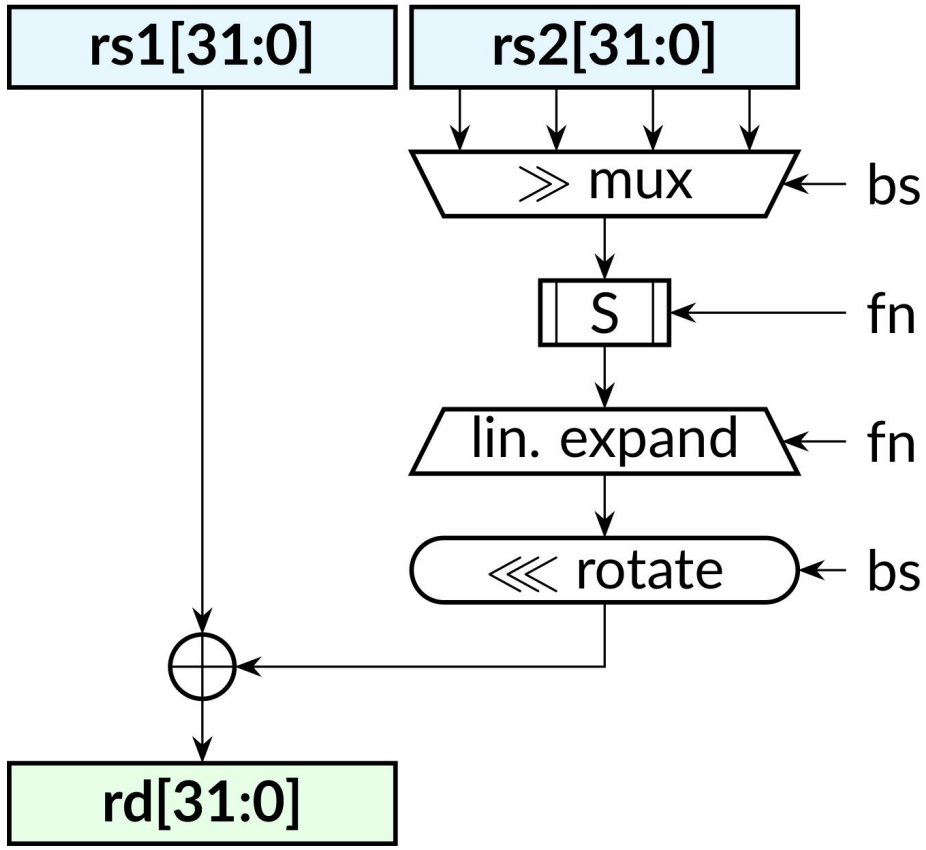
- Like any 800-90B module, the entropy source has mandatory startup and health tests, which may put it in **DEAD** state.
- For AIS.31 some of its startup and online tests may be implemented in software; the driver must be in module certification scope.
- $H \geq n + 64$ with $H=128$, $n=192$ is from Appendix. A of draft 800-90C.
- If the ES module can't meet the 0.75 min-entropy rate, either it needs to be reconfigured (e.g. sample rate) or have a better conditioner.

For more rationale, see the spec and https://eprint.iacr.org/2020/866

Implementing the DRBG Drivers

Use the RISC-V Scalar Crypto Extensions!

- The 32-bit and 64-bit AES and SHA instructions in scalar crypto allow *fast, lightweight, constant time AES DRBGs*.
- Reduced overall size and power to having an AES core just for “random”.
- The vector crypto extension will have even higher output speeds for DRBGs.
- Also supports new / non-standard RBGs.



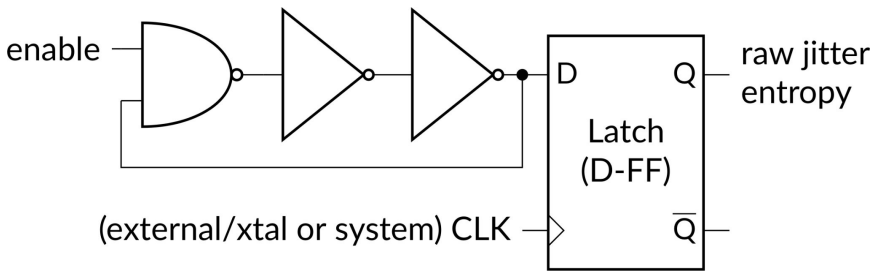
Outline: The RISC-V Entropy Source

1. Cryptography/Driver Developer: The RISC-V ISA Specs.
2. **Chip Vendor: On Integration of ES Modules & RISC-V CPUs.**
3. Security testing lab: Raw sample interfaces (“GetNoise”).

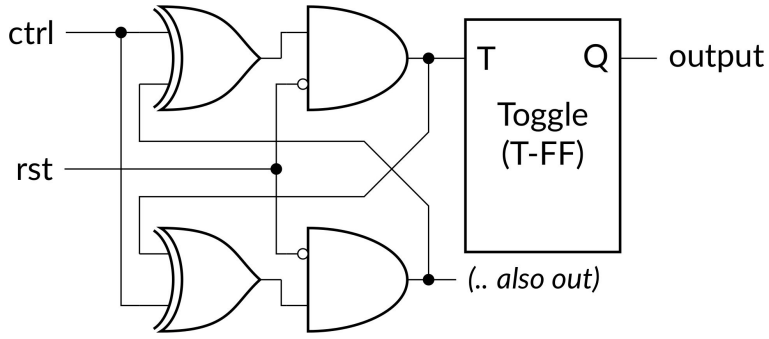
Physical Noise Sources

Need well-understood Entropy: Stochastic Models etc.

- RISC-V ES spec allows any physical noise source suitable for 90B or AIS-31.
- Note: Post-Quantum Crypto (PQC) doesn't need "quantum" noise sources.
- **Ring Oscillators** are common choices. Can be built with standard cell libraries: Cheap and small area (as little as few thousand NAND2 gates).



A ring oscillator with 3 inverters in a free running loop.



A temporarily oscillating "metastable" TERO configuration.

Separate Entropy Validation Scope

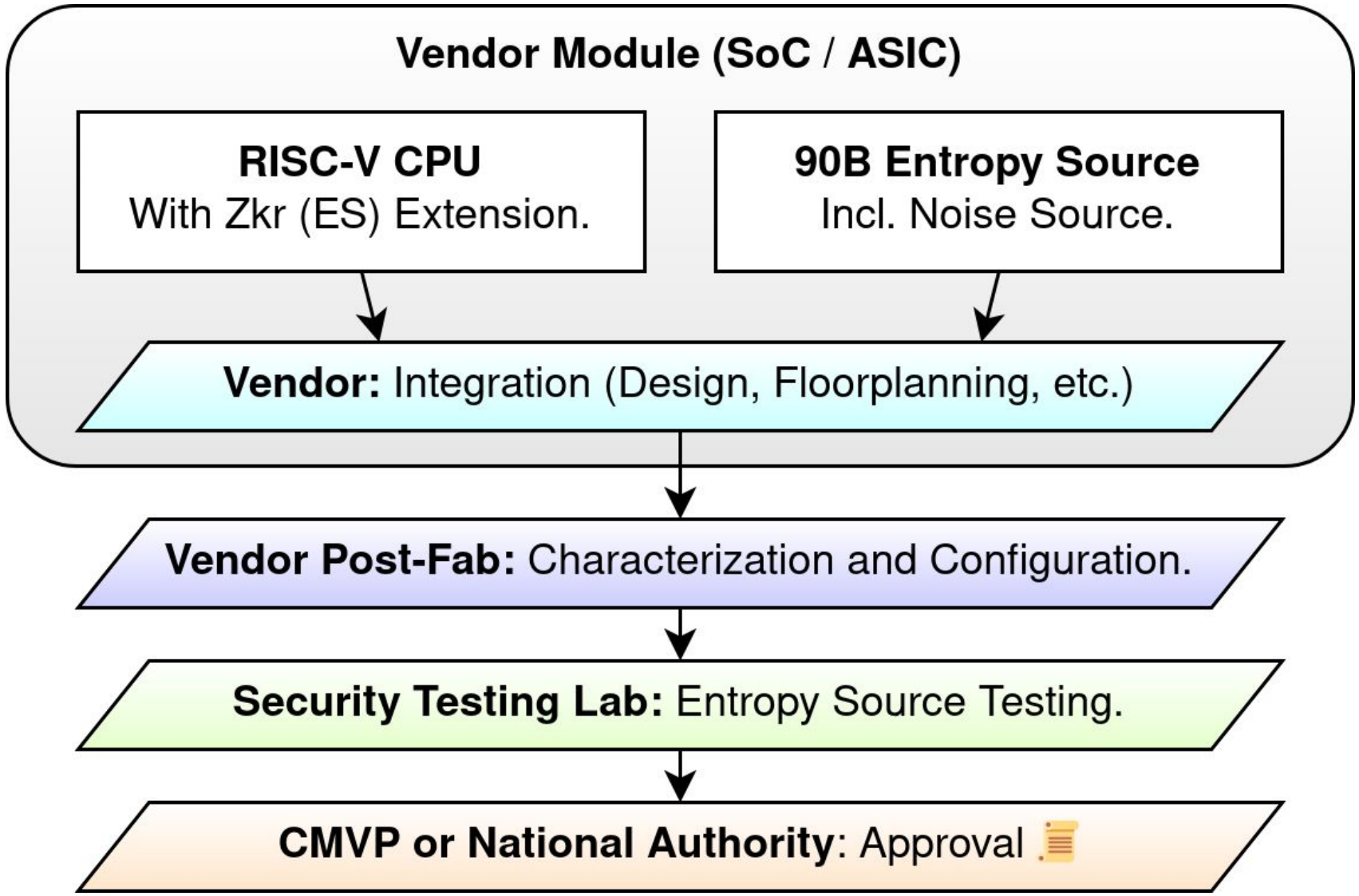
Design reuse little easier for new 90B validations

- (RISC-V) CPUs and Entropy Source modules are licensed by vendors and put into a silicon chip with a bunch of other stuff.

- **Goal**: 90B Entropy Source Module (or AIS-31 RNG IP) can be licensed from a 3rd party and integrated with a RISC-V core.
  -> Thanks to the RISC-V ES Spec, firmware knows what to expect.

- The new *"Entropy Source Validation Scope"* hopefully allows 90B validations to be partially re-used for chips with RISC-V CPUs.

RISC-V + Entropy Source Integration



RISC-V Entropy Source Integration

Checklists 1: Processor Design and Verification

Verify that the RISC-V ISA-defined behaviors are correctly implemented.

Here ES is an abstraction - a *smoke test ES module* may be helpful. Examples:

- ✅ ES16 Entropy words can only be read once (no “peeking” into entropy).
- ✅ Invocation of start-up and online tests on reset or when appropriate.
- ✅ State transitions, especially in case of fatal and non-fatal error signals.
- ✅ Access control in User (U), Supervisor (S), and Machine (M) mode.
- ✅ Custom GetNoise / “mnoise” test interface does not have undue access.

RISC-V Entropy Source Integration

Checklists 2: Noise Source Design and Analysis

The Noise Source Produces the required amount of entropy on the target chip.

- ✅ A stochastic model (or a heuristic argument) that explains why the noise source output is from a random, rather than pseudorandom (deterministic) process.
- ✅ A complete physical model is not necessary, but entropy needs to be justified on the technology node (Semiconductor process? Temperature? Freq? etc.)
- ✅ The noise source is not externally observable (e.g., strongly correlated with the time of day, external power or emissions, or otherwise public or predictable).

Example: For a *ring oscillator* one could show that the source derives an amount of physical entropy from local, spontaneously occurring Johnson-Nyquist thermal noise.

RISC-V Entropy Source Integration

Checklists 3: Entropy Source Module Design

Verify that Entropy Source meets all of the Standard Requirements

A compliant entropy source consists of a noise source *and* additional components whose purpose is to guarantee consistent entropy output.

- ✅ Start-up and health tests. Vendor-defined tests.
- ✅ The conditioner (cryptographic/vetted or non-vetted).
- ✅ A security boundary with clearly defined interfaces. Etc..

One can tabulate all the **SHALL** statements from SP 800-90B, FIPS 140-3 IG (or BSI AIS-31). https://github.com/usnistgov/90B-Shall-Statements

RISC-V Entropy Source Integration

Checklists 4: Characterization and Configuration

Configure source, perform 90B entropy estimation (or AIS-31 statistical tests).

ES modules may have parameters that can be configured at factory (after fabrication).
Example parameters: Ring oscillator sampling Interval, health test α and β bounds.

- ✅ Perform ES Module-defined characterization procedure to derive configuration.
- ✅ Config and parameters must match those submitted to the Security Testing Lab!

Integration testing also replicates the Entropy Source Validation Test System (ESVTS):

- ✅ Use the (IID/Non-IID) estimation algorithms: Record entropy source and raw noise samples using the vendor-defined custom interface. Restart tests, etc.

Outline: The RISC-V Entropy Source

1. Cryptography/Driver Developer: The RISC-V ISA Specs.
2. Chip Vendor: On Integration of ES Modules & RISC-V CPUs.
3. **Security testing lab: Raw sample interfaces (“GetNoise”).**

Suggested GetNoise Test Interface

A Custom CSR location

- Access to raw samples is required for SP 800-90B validation -- the GetNoise conceptual test interface defined in Section 2.3.2 if 90B.

- It is suggested that a custom M-mode **mnoise** CSR = GetNoise().

- *Custom*: Address is vendor-specified. The interface may be permanently disabled in production chips (after factory tests).

- Due to variance in entropy sources, only one CSR bit is defined; NOISE_TEST. However much of the simple testing scripts can be shared between different CPU & ES implementations (open source!)

Suggested GetNoise Test Interface

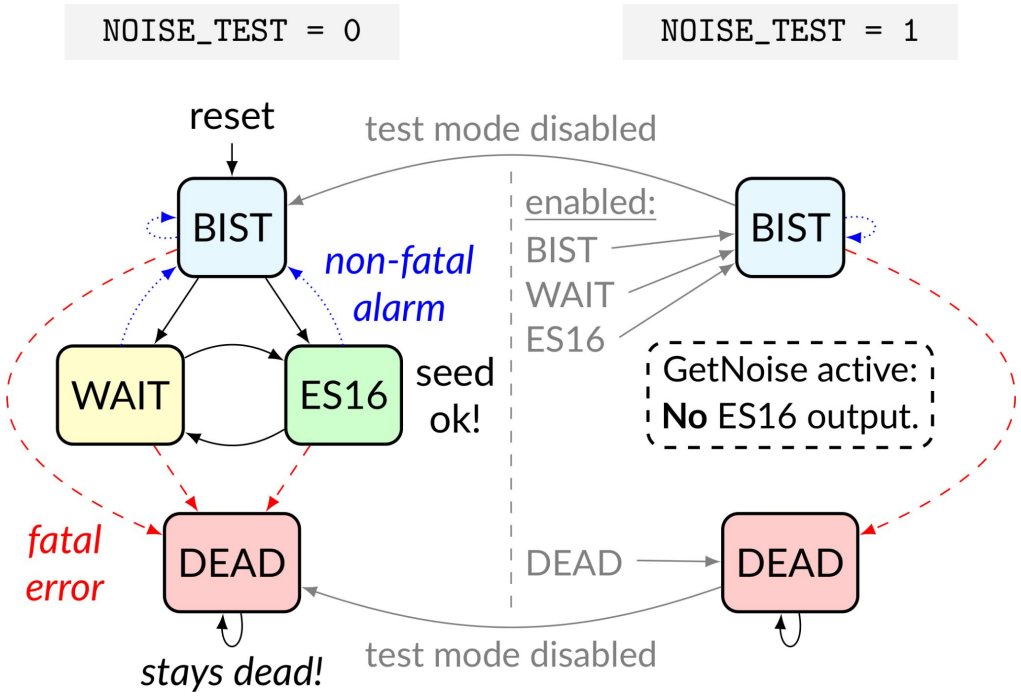
A Custom CSR location

Due to security concerns **seed** CSR entropy is not available while the source is in **mnoise** test mode (*bypassing health checks and conditioning!*)

Testing lab runs scripts that gather datasets (raw, conditioned, startup) and performs estimations.

Note:

The **mnoise** CSR is probably used for the vendor entropy characterization & configuration step too.



Thank You!

Questions..

riscv.org

pqshield.com

Link to specifications:

https://github.com/riscv/riscv-crypto/releases