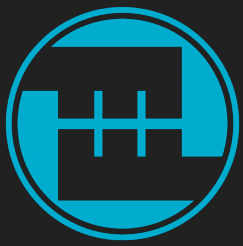
(Current) Limitations on SP 800-90C External Conditioners and How to Overcome (Some of) Them

Joshua E. Hill, PhD

20260422
ICMC N20b

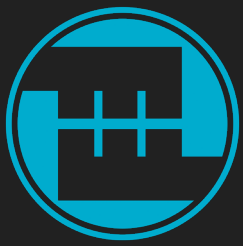
What the What, Now?

- We look at an SP 800-90C restriction on the Get_entropy_bitstring() interface and highlight some consequences of this restriction to common design patterns.
- We present a design that resolves many of these issues.
- We discuss some relevant parts of CMVP's recently proposed SP 800-90C IG.



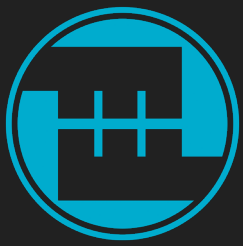
The Get_entropy_bitstring() Restriction

- The Get_entropy_bitstring() interface is the only way that 90C constructions get entropy they can credit.
- Get_entropy_bitstring() has some restrictions (90C Section 3.1, Requirement 1)
  - “The Get_entropy_bitstring process **shall** only be used to access one or more **validated** entropy sources. Any non-validated entropy sources **shall** be accessed by a separate process to avoid possible misuse.”



The Get_entropy_bitstring() Restriction

- Get_entropy_bitstring() is the only input into SP 800-90C external conditioners.
  - There is no *additional_input* analog in this setting so there is no way to integrate input any non-validated ESs at this level.
- External conditioning is used any time that the Get_conditioned_full_entropy_input() or Get_conditioned_input() processes are invoked.
- DRBGs have the ability to integrate inputs from non-validated ESes as a *personalization_string* (on instantiation) or *additional_input* (on generate and/or reseed).

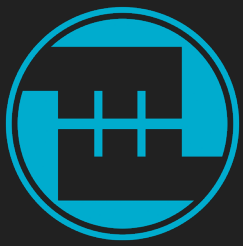


The Get_entropy_bitstring() Restriction

- Many situations exist where a single source base or binary is used across multiple OEs.
- The validation status of the OEs' ESs varies, and there is little that vendors can do about that.
  - E.g., Intel and AMD RdSeed certs apply to very specific chips.
    - These designs are very commonly available but are only intermittently ESV validated.
  - Many such designs are well-engineered physical noise sources and when such ESs are available their output ought to be integrated into the DRBG internal state as a matter of good engineering practice, even if their output cannot be credited as contributing entropy.

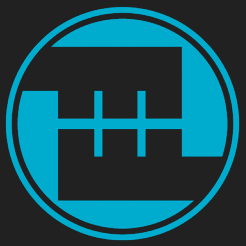
The Get_entropy_bitstring() Restriction

- JEnt is commonly used in OEs where the underlying hardware ES may or may not be validated.
- Modern JEnt is mostly straightforward to validate for specific hardware, but…
  - It is a classic non-physical (found) source and developing a conservative entropy bound is fraught.
  - The resulting validations are hyper-hardware and configuration specific.



The Get_entropy_bitstring() Restriction

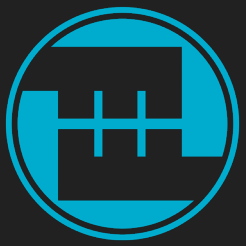
- To account for the shifting ES validation status across multiple OEs / implementations, there should be some way to combine the contributions of validated and non-validated ESs.
  - The validation status of each ES may not be known by the implementation *a priori*, but…
  - Validation status of these ESs for all evaluated OEs will be known and verified during the 90C (and FIPS) validation.
- It isn't clear what we gain by precluding non-validated ESs from being used by Get_entropy_bitstring(), so long as this call necessarily returns the requested entropy from a validated ES.



The Get_entropy_bitstring() Restriction

The Get_entropy_bitstring() validated ES restriction has the following consequences:

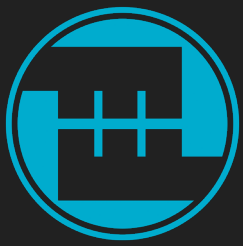
- It essentially prevents the use of external conditioners to combine multiple ESs when the validation status of those ESs vary across OEs / implementations.
    - This forces the use of configuration parameters that govern how data is routed.
- Vendors seeking ease of validation commonly exclude reasonable ESs for transient validation status reasons.
    - This effectively incentivizes brittle designs to reduce validation effort.
- Vendors may use DRBGs (which are allowed to combine validated ES data and other data), but…



DRBG Processing of Personalization String / Additional Input

The protections against malicious input provided by the DRBG when combining the internal state / *entropy_input* and *personalization_string* / *additional_input* DRBG parameters seem inconsistent.

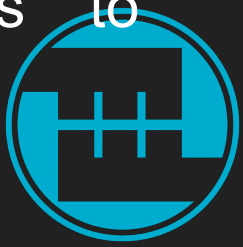
- Mostly this data is concatenated then cryptographically processed (via Update / Derivation Function / Hash).
- The CTR_DRBG with no df directly XORs this data into the DRBG internal state.
  - See step 4 of the CTR_DRBG_Update process in SP 800-90A Section 10.2.1.2.
  - This seems to give some classes of attacker abilities like those they have for CBC-MAC-style conditioners taking input from both trusted and malicious sources.



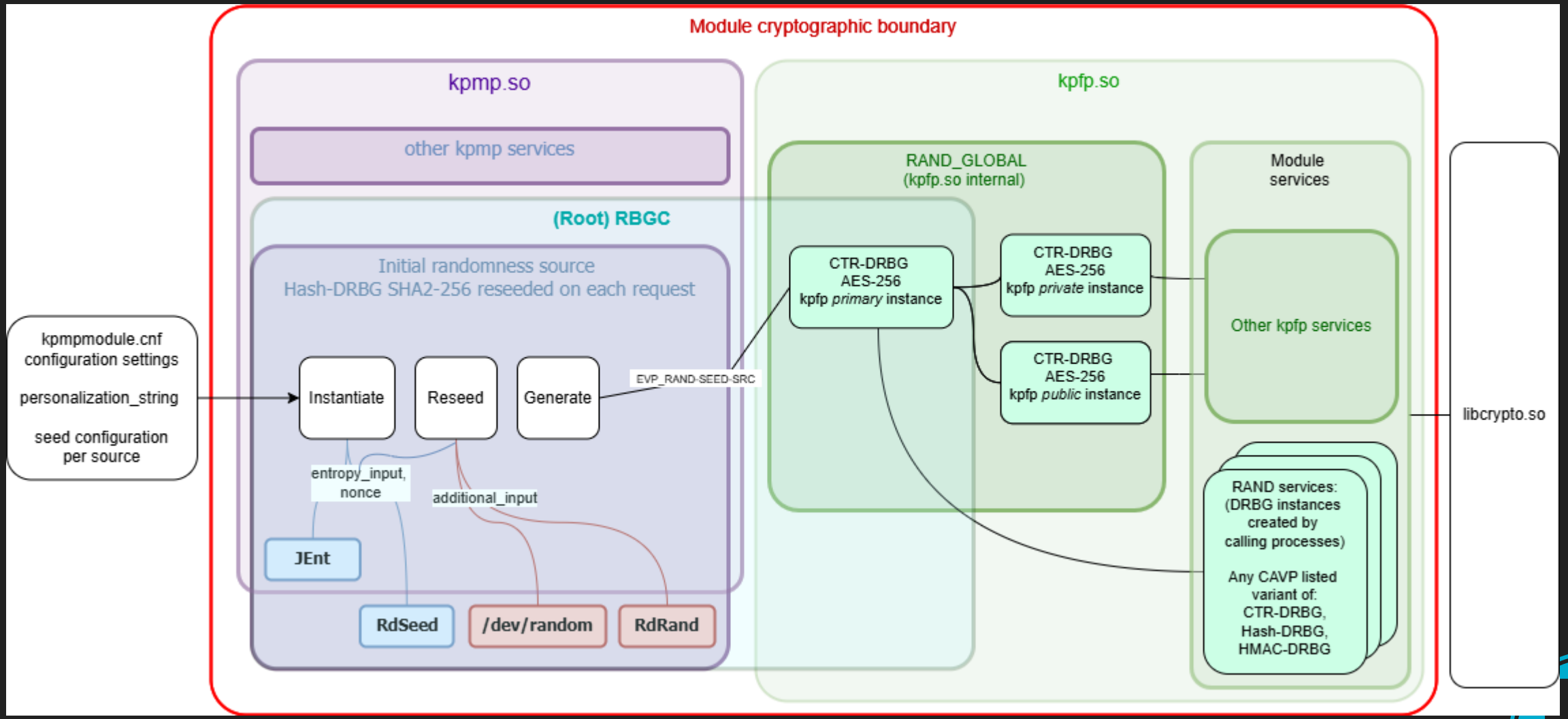
One Approach

Produce a solution that:

- Offers SP 800-90A/B/C compliance for a full RBGC construction.
  - Addresses the Get_entropy_bitstring() non-validated entropy source exclusion issue.
- Is easily integrated with existing libraries:
  - Many libraries (and their users) presume that all seeds are full entropy.
  - Most libraries lack a centralized infrastructure for distributing *additional_input*.
- Is configurable to support multiple OEs.
  - Make the implementation invariant and push OE-specific details to configuration.



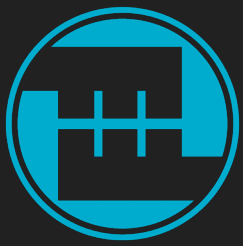
Our Proposed Solution



The RBGC IRS RBG

The RBGC Initial Randomness Source (IRS) RBG is central to the RBG construction. To address these goals, it should:

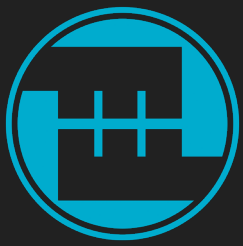
- Output full entropy strings.
- Integrate input from all available healthy ESs, both validated and non-validated.
  - Compliantly segregate the output of validated ESs from the output of other ESs.
  - Use a conservative DRBG design that cryptographically integrates *additional_input*.
- Degrade gracefully in the event of failures.



The RBGC IRS RBG

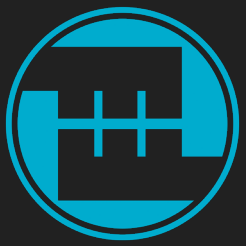
The RBGC Initial Randomness Source (IRS) RBG implemented here:

- Uses a Hash_DRBG with a security strength (s) of 256.
- Is an RBG that is (or acts “much like”) a fault-tolerant RBG3(RS).



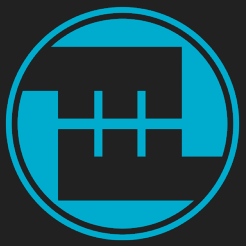
RBGC IRS RBG Goals

- This RBGC IRS DRBG reseed cadence is the same as for the DRBG in an RBG3(RS).
  - Every generate is preceded by a (RBG3(RS)-Method-A style) Reseed.
    - s+64 (320) bits of entropy is requested from **each** of the healthy validated ESs.
    - The *seed_material* for the Reseed is the concatenation of the entropy strings produced by the healthy validated ESs.
  - The RBGC IRS RBG also uses similar amounts of information from the non-validated ESs as *additional_input* for the Reseed.
- The RBG3(RS) DRBG output size restrictions are enforced.
  - The RBGC IRS RBG DRBG Generate outputs at most s (256) bits.



RBGC IRS RBG Validation Characteristics

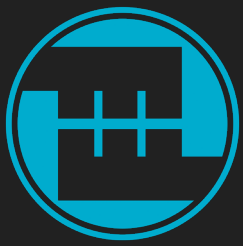
- When physical ESs are available, the RBGC IRS RBG can be configured to use Method 1 entropy counting.
  - The RBGC IRS RBG is then an RBG3(RS).
  - Method 1 counting enforces the use of at least one healthy physical ES for instantiation and for all reseeds.
- The RBGC IRS RBG can also be configured to use Method 2 entropy counting.
  - The RBGC IRS RBG is then an RBG2(NP).
  - This is the required configuration when the only validated ESs available are non-physical.



Recently Proposed 90C IG Updates

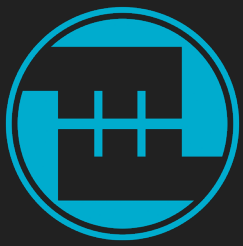
The proposed 90C IG includes a possible relaxation of this Get_entropy_bitstring() interface requirement for RBGCs which applies when:

1. At least one entropy source is validated; entropy can only be credited from validated entropy sources.
2. The full amount of entropy of entropy is requested from each entropy source.
3. If any entropy source reports an error, the request must report a failure.
4. The outputs of all the sources are concatenated.



Proposed Resolution

- This approach is limited to RBGCs:
  - It isn’t clear why this could not apply to all uses of this interface.
  - This approach doesn’t address the broader external conditioning issue.
- This approach doesn’t support error-tolerant behavior.
  - Any of the ESes could be the one approved source, so any ES failure must result in an overall failure.



What the What, Then?

- We looked at an SP 800-90C restriction on the Get_entropy_bitstring() interface and highlight some consequences of this restriction to common design patterns.
- We presented an RBG3(RS)-like IRS for an RBGC that overcomes these issues.
- We discussed some draft CMVP guidance that may resolve some of these issues in some settings.