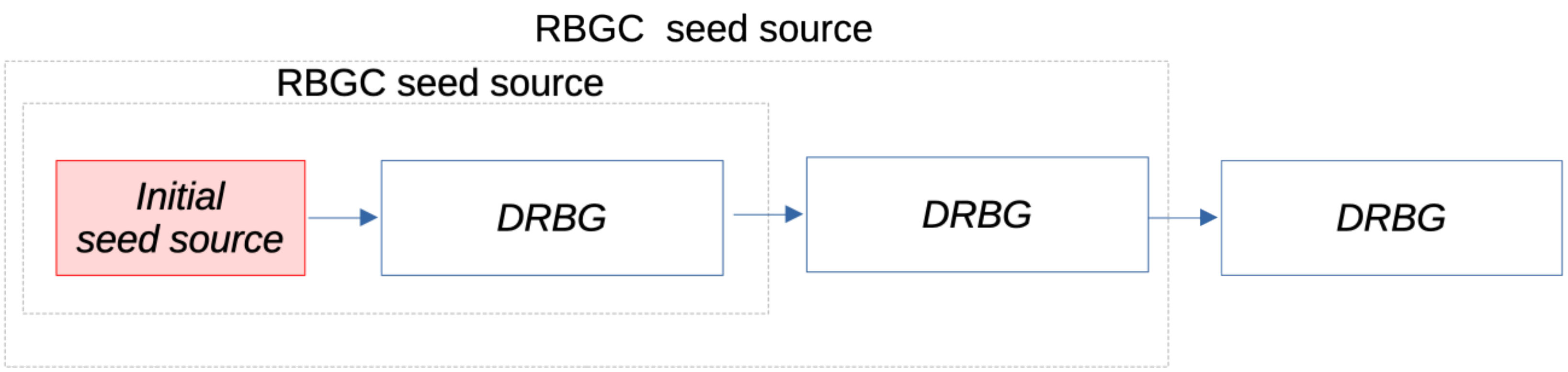
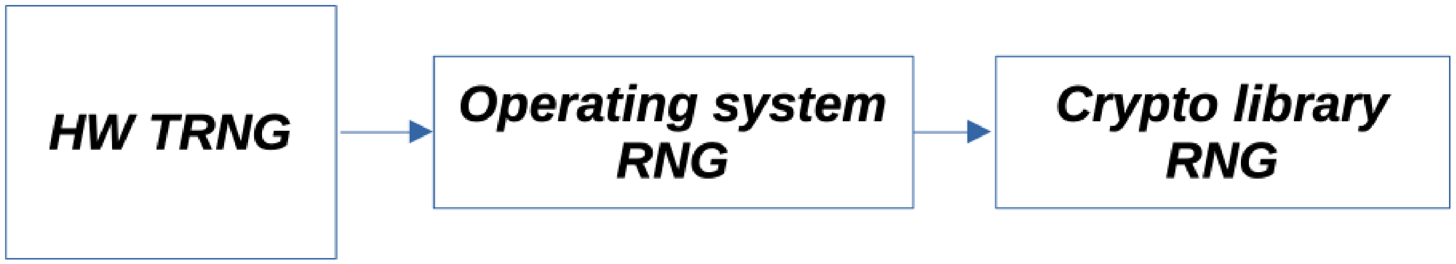


RBGC: Chains of RBGs

John Kelsey, NIST and COSIC/KU Leuven

Disclaimer: all this is still in discussion and subject to change

Chains of RBGS are common in software

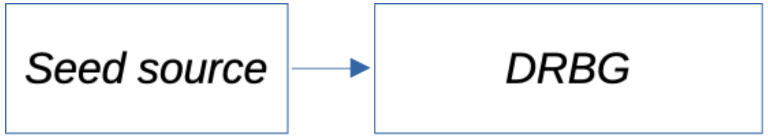


- Hard to see how else to do it

- 90C needs to allow this

  … without making the standard too complex to understand

Solution: RBGC construction



RBGC consists of:

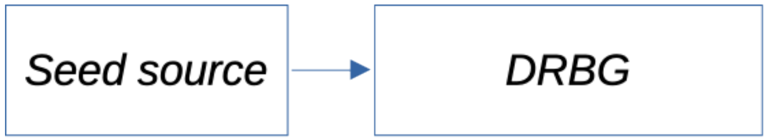
- DRBG
- Seed source
  … which can be another RBGC

RBGC components

Seed source

Any of

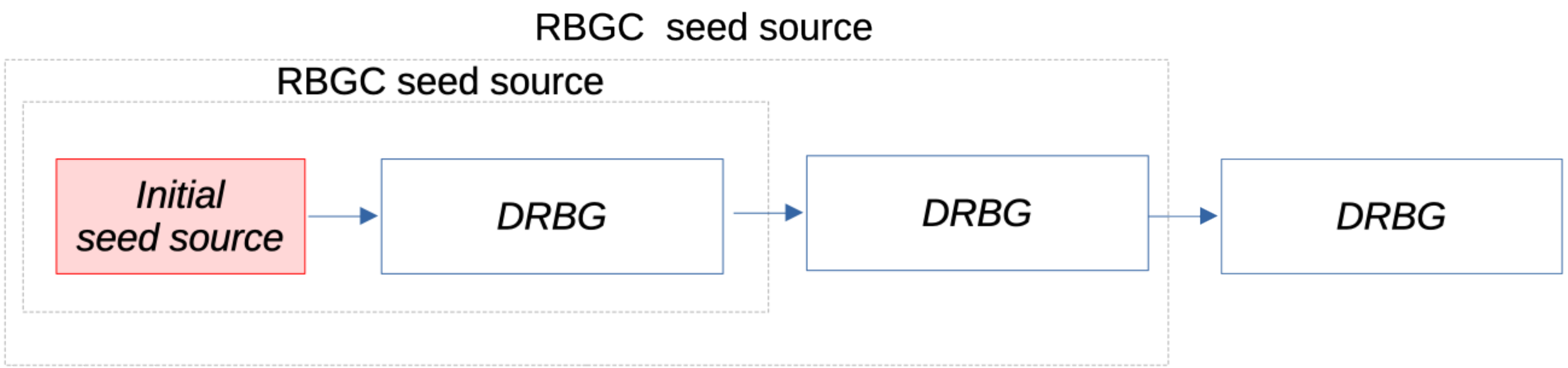
- RBG2
- RBG3
- Full entropy source
- Another RBGC ← *this allows chains and trees of RBGCs*



DRBG

Any approved DRBG

The initial source

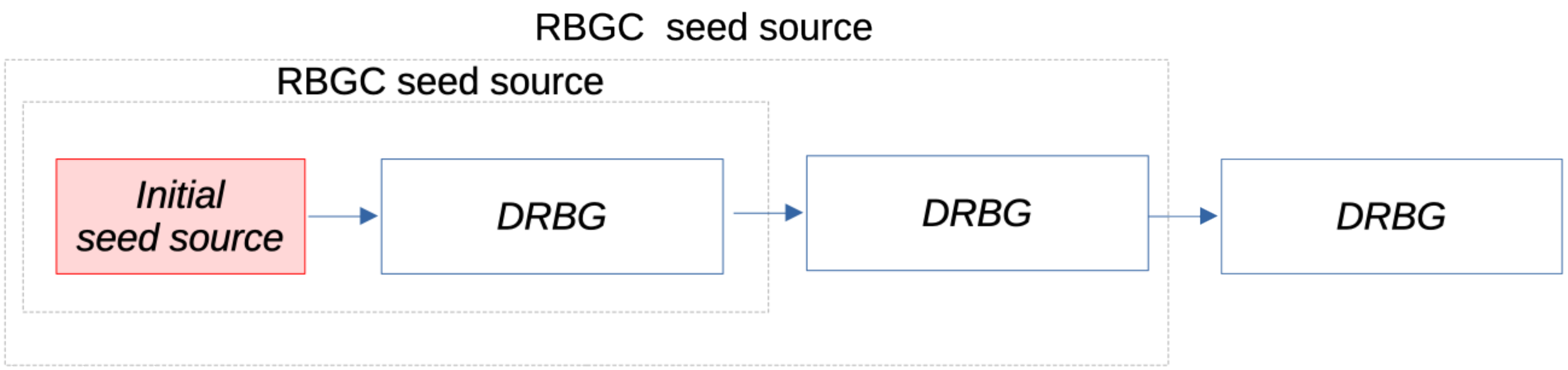


An RBGC can be a seed source…

… but the initial seed source has to provide some entropy

- RBG2
- RBG3
- Full entropy source

Everything depends on initial source



Requirements:

- Available entropy source
- Strong output bits

Initial seed source may be:

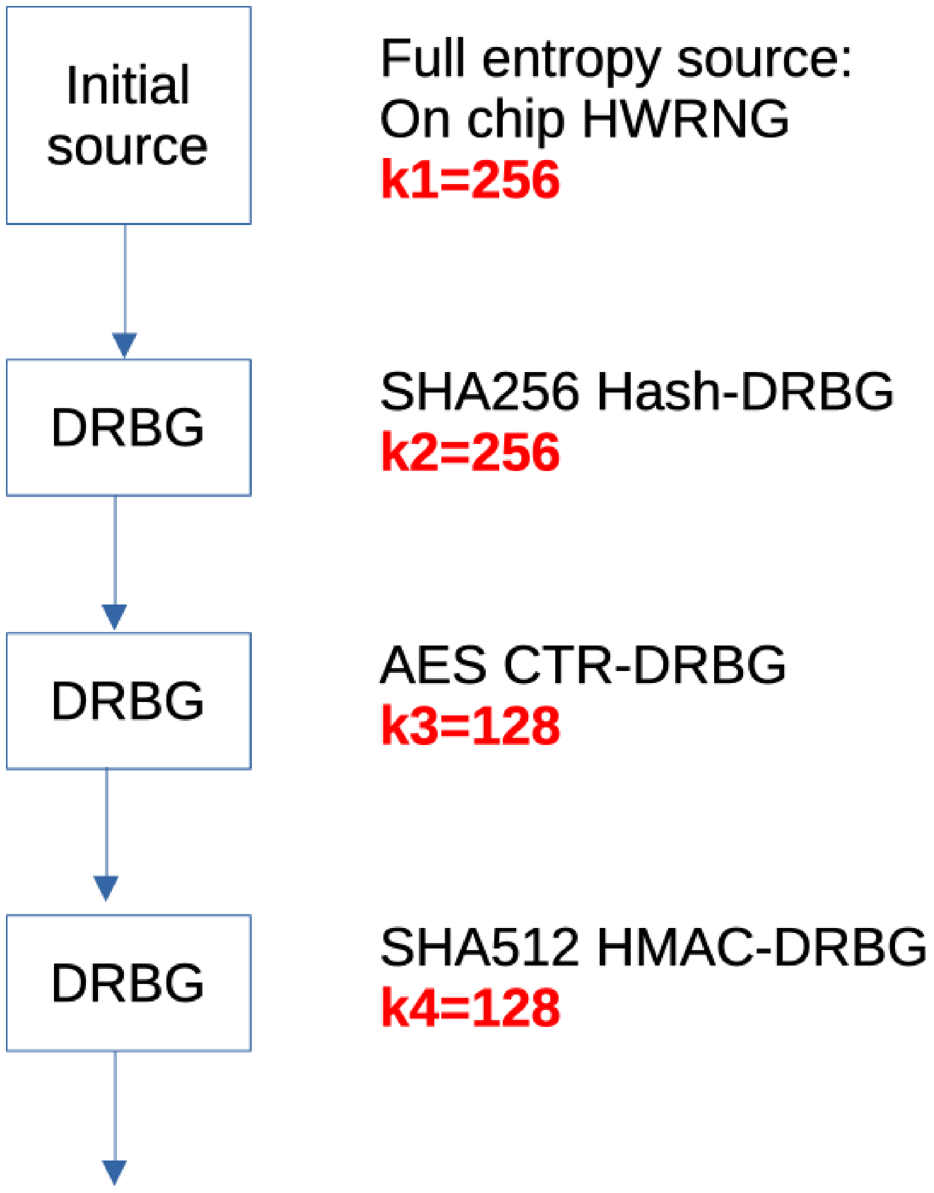
- RBG2(P)
- RBG2(NP)
- RBG3
- Full entropy source

Chains of DRBGs: requirements

- Security strength can’t increase
- Each DRBG has exactly **one** seed source
  - May also incorporate additional input
- A DRBG may provide seed material **AND** random bits
- **No loops allowed (See below)**

Example chain

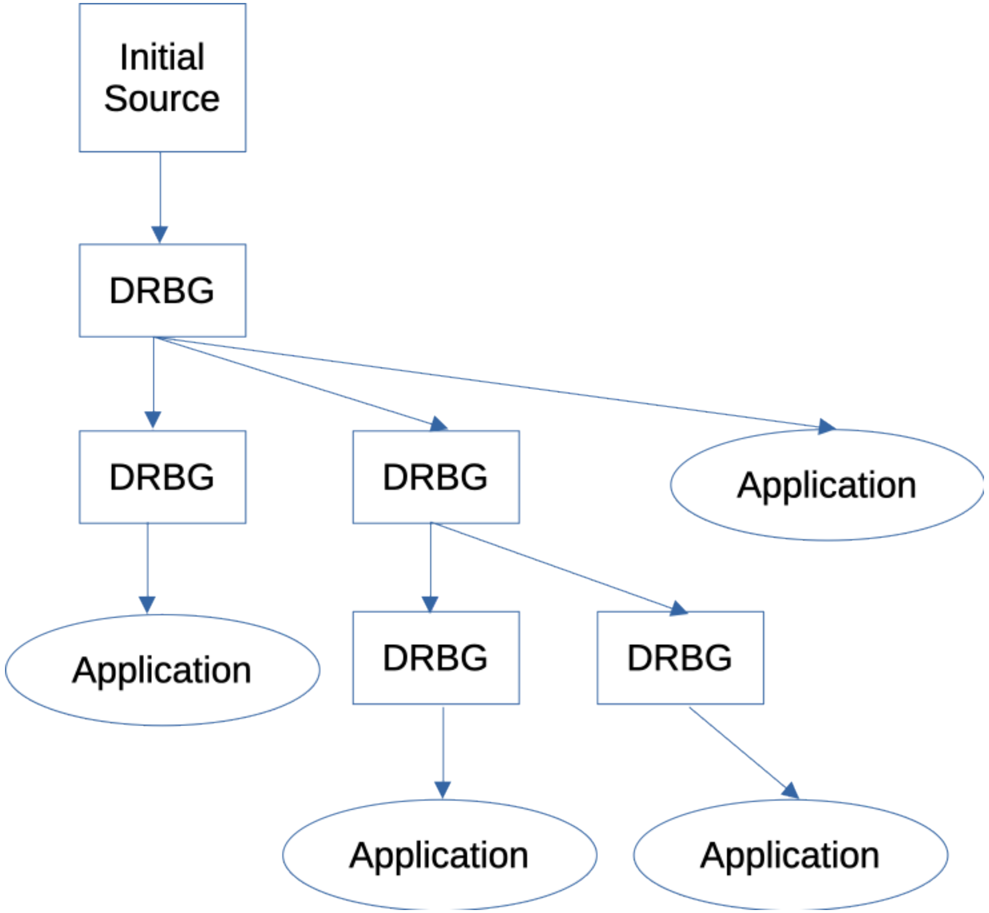
- Each DRBG has one seed source
- Security strength can't go up



Trees of RBGs

- Each DRBG: one seed source
  - May also incorporate additional input
- Initial source: Ultimate root of trust
- One source → many DRBGs
- DRBGs can provide seed to other DRBGs

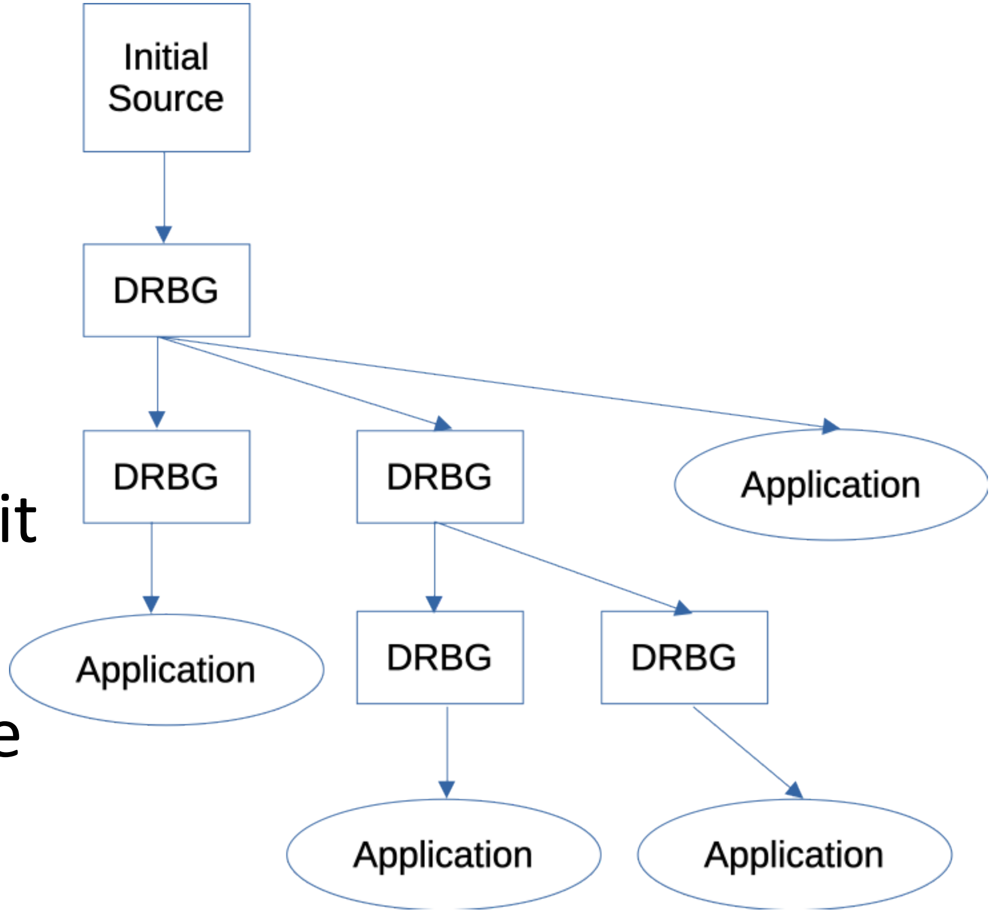
  …AND random bits to applications



No limit on DRBGs in chain or tree

Should there be?

- Hard to justify any particular number
- Not a clear security reasons for specific limit
- Not sure what limit of useful designs will be



Ancestors, descendants, and loops

For each DRBG in RBGC:

- Ancestors

    Everything in chain that seeds DRBG

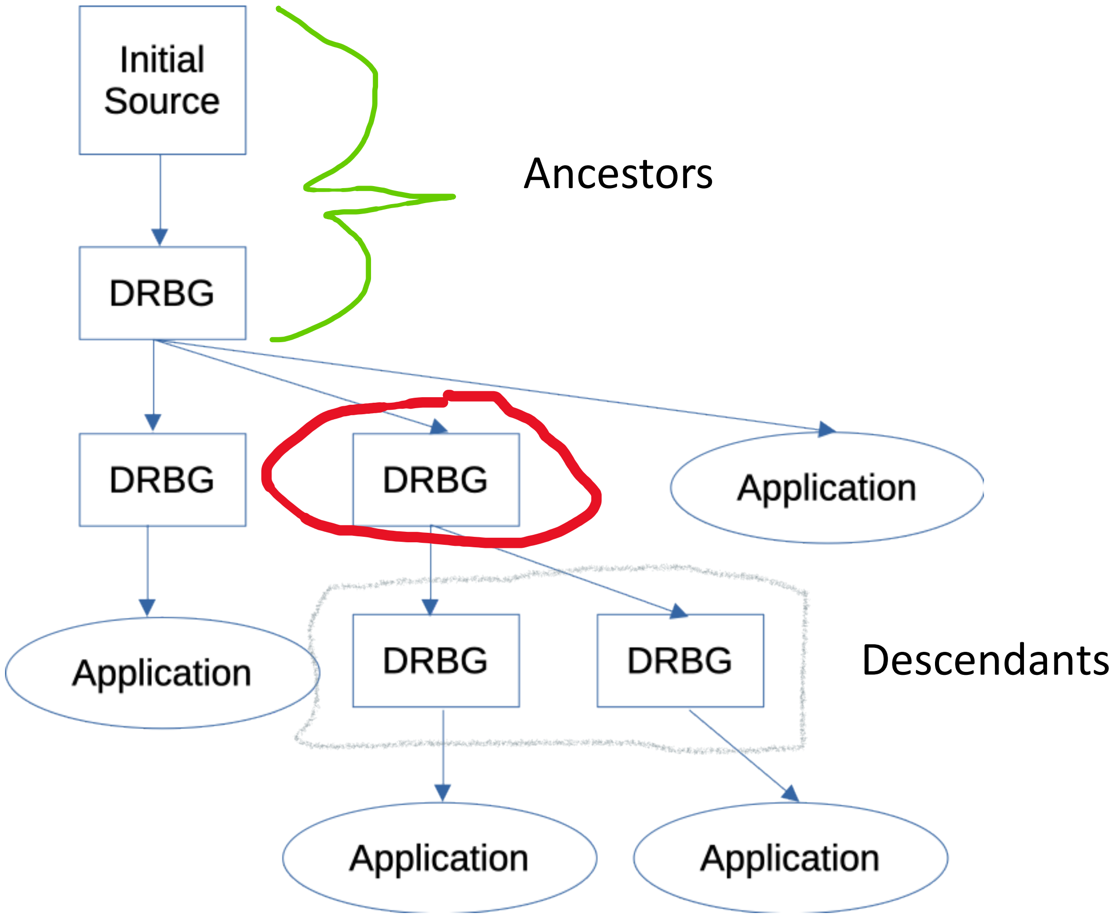
- Descendants

    Everything in tree seeded from this DRBG

- Loop

    Any DRBG is its own ancestor

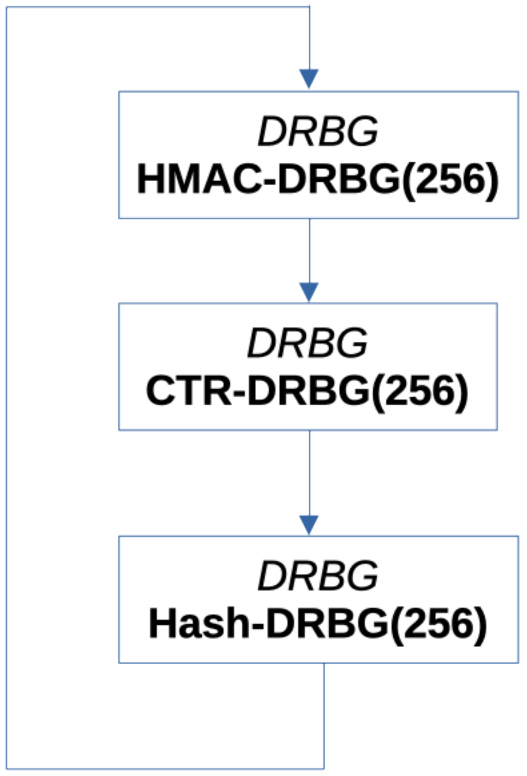
    Not permitted (for obvious reasons)



Loops are not allowed

Example: RBGC with loop

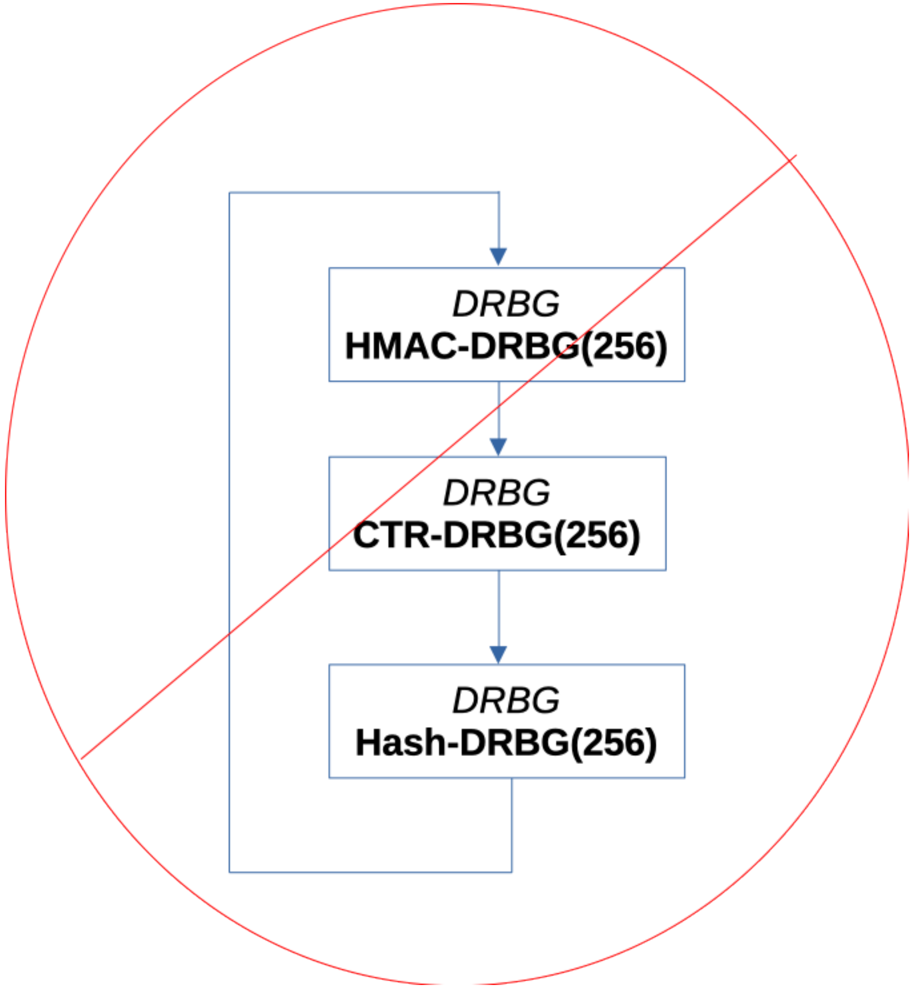
- No entropy, no security
- Every DRBG is its own ancestor



Loops are not allowed

Example: RBGC with loop

- No entropy, no security
- Every DRBG is its own ancestor

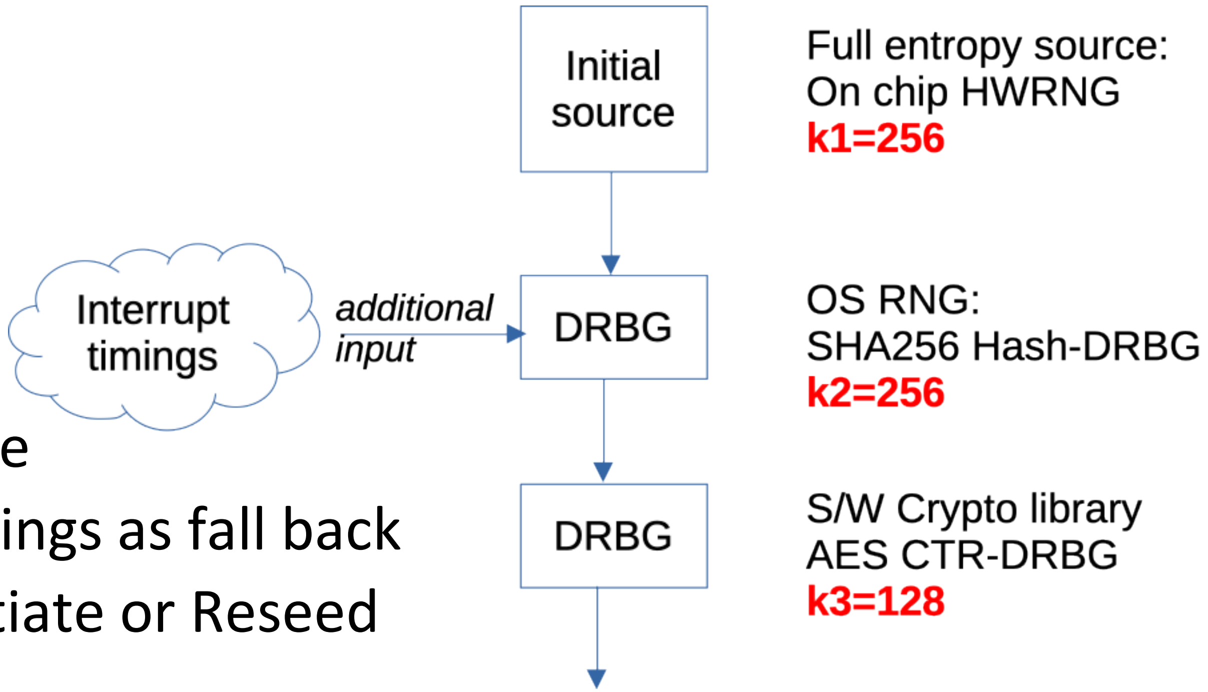


This is insecure, and not allowed in 90C

What about additional input?

- DRBGs can incorporate additional input
  - Instantiate, Reseed, Generate
- Not many requirements on additional input
- Additional input is allowed for DRBGs in an RBGC construction
- DRBGs should not make a loop using additional input
  - Not a security issue, but seems like a bad idea

Example: How to use additional input?

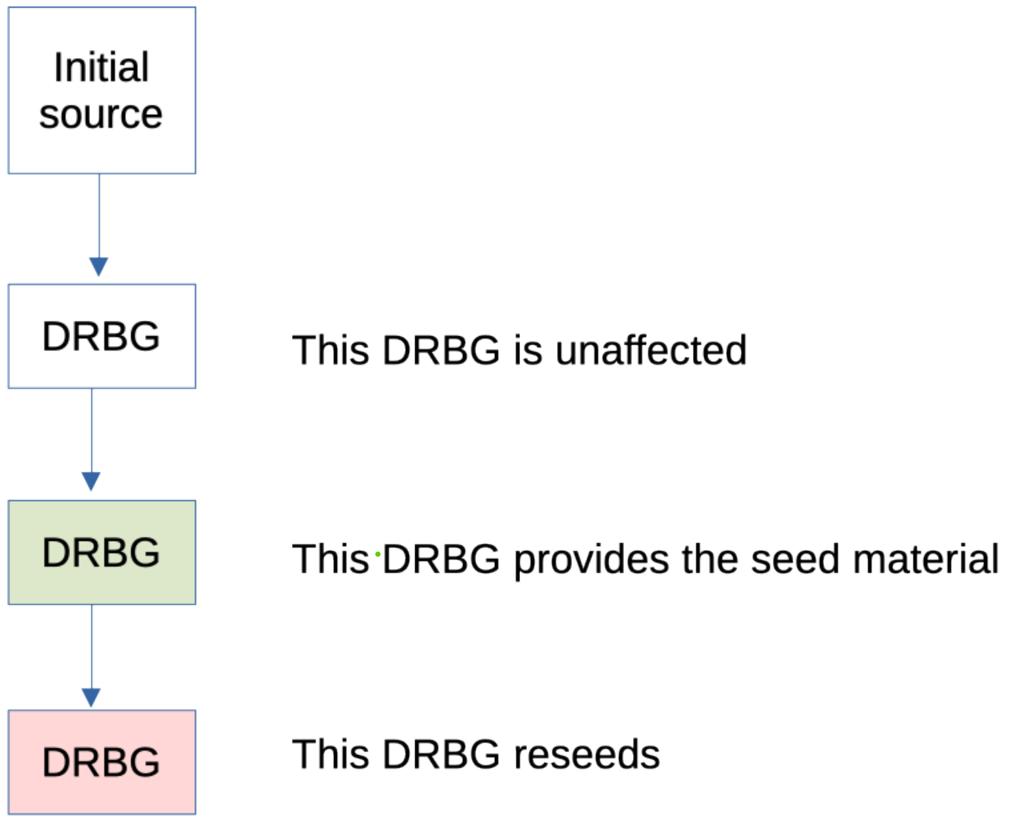


- OS RNG:
  - Use hardware TRNG as seed source
  - Collect entropy from interrupt timings as fall back
  - Put into additional input of Instantiate or Reseed

Could design RNG to instantiate only when sufficient entropy from interrupt timings

Reseeding a DRBG in RBGC construction

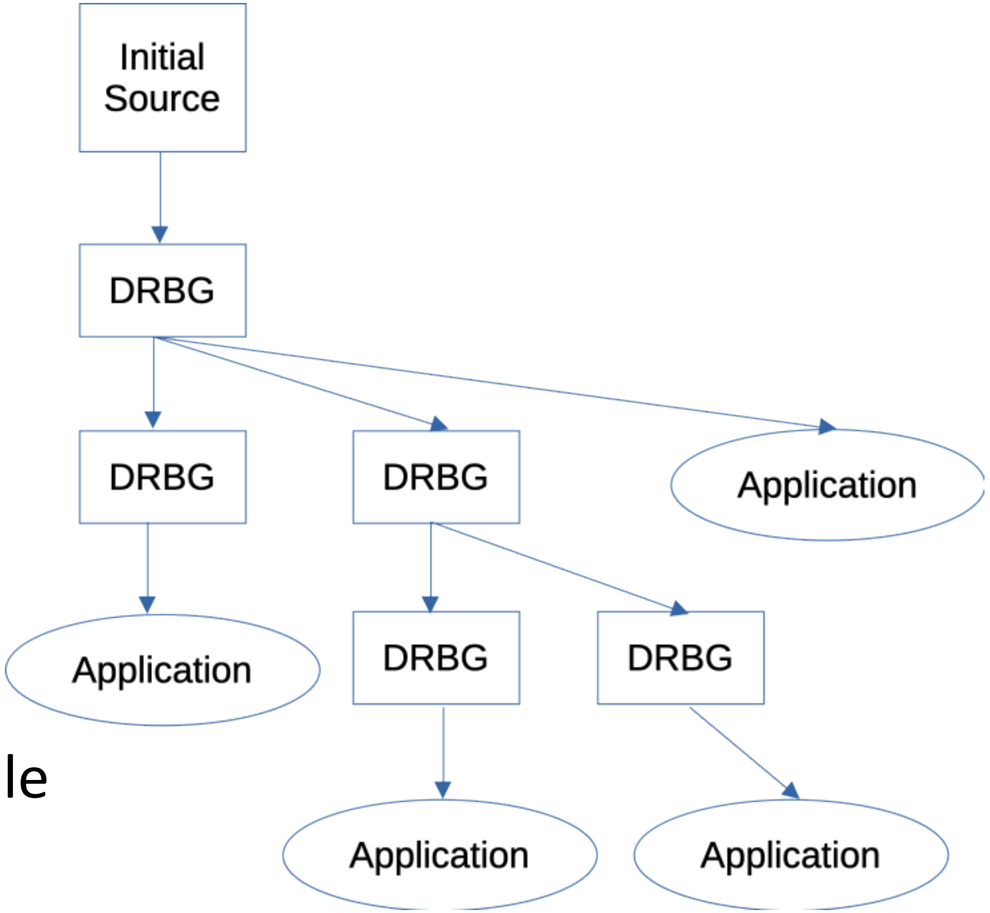
- Reseed draws new seed material from seed source
- DRBGs SHALL support reseed request
- Seed source provides requested amount of seed material
- Reseed DOES NOT recurse up the chain



No guarantee of fresh entropy from reseed.

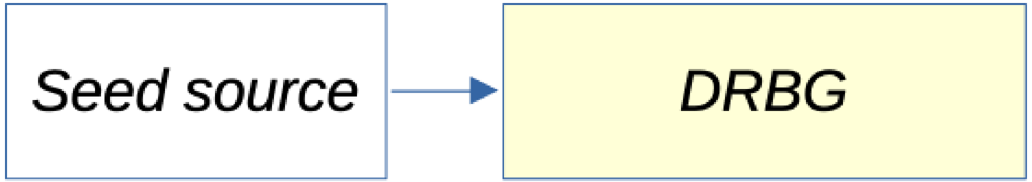
Prediction resistance is not supported

- DRBGs in RBGC do not support prediction resistance
- Avoid potential for denial of service from too much demand on initial source
- RBG2s do not always reseed on demand
  - Sometimes reseed as entropy becomes available



Hard problem: Modularity

Common for software to be written without knowledge of what else will be on platform.



Problem: how does lab evaluate RBGC in software module without seeing seed source?

This is extremely common situation. How should we deal with it?

Suppose we're given this DRBG

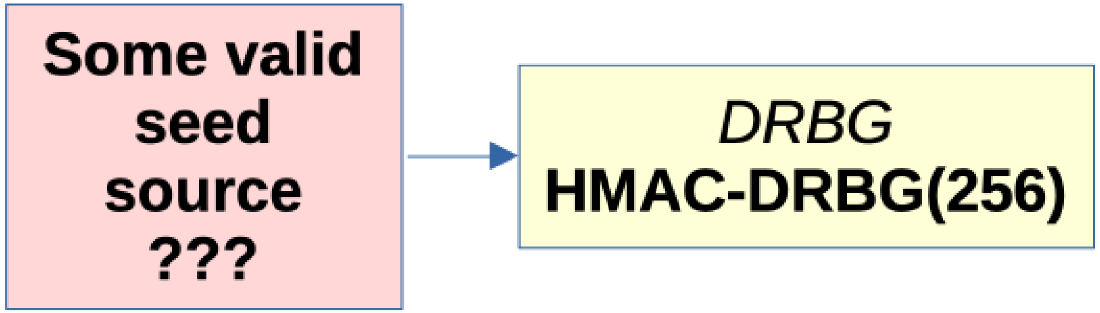
Claim: RBGC

For this to be a valid RBGC:

- Seed source must be one of
  - RBG2
  - RBG3
  - Full entropy source
  - RBGC
- Seed source security strength

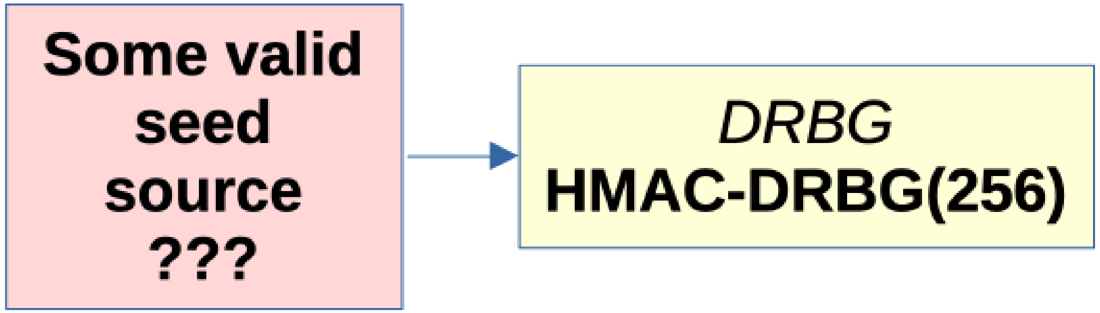
  At least as strong as DRBG
- Seed source must not rely on DRBG for seed

  No loops



Lab can't evaluate what it can't see

How should we approach this?



Obvious approaches:

1. Validate DRBG, specify requirements for seed source in certificate, hope for the best

2. Validate as DRBG only. To get validation as RBGC, require it to be combined with seed source meeting requirements.

Wrap up

- RBGC construction allows chaining of DRBGs
  - Permits chains or even trees
- Initial source has live entropy source
  - RBG2, RBG3, full entropy source
- No prediction resistance, yes reseeds
- Hardest problem: modular evaluation